Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN TECNOLOXÍAS DA INFORMACIÓN

Aplicación móvil para predicciones de apuestas deportivas.

Estudante: Alejandro Batista Díaz
Dirección: Miguel Rodríguez Penabad

A Coruña, setembro de 2020.

Dedicatoria

A toda mi familia y en especial a mi abuela materna.

Agradecimientos

Quiero agradecer a Miguel R. Penabad la paciencia que ha tenido conmigo en la elaboración de este proyecto, a la vez que a todos los profesores de la facultad de Informática de A Coruña por el tiempo que me han dedicado, con mención especial a Javier París, Fran Novoa, Sonia Suárez y Mon López. Tuve también la suerte de conocer a Víctor Gulías y compartir con el algún café en la barra de la facultad mientras Lázaro hacía un discurso de los suyos.

Resumen

El objetivo de este trabajo final de grado es la creación de una aplicación móvil que prediga resultados de apuestas deportivas de La Liga Española de fútbol.

Para llevarla a cabo se realizará un proceso completo que inicia en la extracción de datos de un proveedor externo y mediante un flujo ETL lo volcará a una base de datos de nuestro diseño y control. Se creará un servicio REST que exporte los datos que requieren las funcionalidades necesarias y finalmente, se diseñará una aplicación móvil donde mostrar las predicciones deportivas en función de modelos estadísticos.

Nos centraremos en las apuestas a número de goles, número de tarjetas y número de córners en un partido.

Abstract

The main objective of this final work of degree is the develop of a mobile application that predicts the sports betting results of the Spanish Soccer League (La Liga).

To make this posible, a complete process must be develop beginning with the extraction of data from an external provider and through an ETL flow, it will be transferred to a database of our design and control. A REST service will be created and it exports the data that require the necessary functionalities and finally, a mobile application will be designed to show sports predictions based on statistical models.

We will focus on betting on number of goals, number of cards and number of corners in a match.

Palabras clave:

- Servicios REST
- ETL
- Apuestas deportivas
- Predicción estadística
- Aplicación Android

Keywords:

- REST Services
- ETL
- Sport Bets
- Statistics and Predictions
- Android App

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introdución

El sector de apuestas está en auge desde 2008 coincidiendo con el principio de la crisis económica. Debido a ello creemos que es un buen nicho de mercado el desarrollo de una herramienta que, mediante cálculo estadístico, pronostique el resultado de apuestas deportivas convencionales.

Nuestro objetivo concreto es el desarrollo de una herramienta software que, obteniendo los datos de un servicio externo, nos proporcione resultados lo mas fiables posibles de las típicas categorías que pueden encontrarse en cualquier casa de apuestas a día de hoy.

Las casas de apuestas ofrecen muchos tipos de apuestas varias, viendo por ejemplo la página web de un conocido broker como es Bet365, vemos que no solo tenemos fútbol, existe un amplio abanico de posibilidades para elegir entre otros deportes.

Las casas de apuestas funcionan de la siguiente forma; para una apuesta concreta, la casa ofrece unos valores (de ahora en adelante nos referimos a ellos como cuotas) que son el número por el que multiplicaríamos el importe apostado en caso de que, al final, esa opción sea la que se produzca en el evento real.

Poniendo un ejemplo con el mismo broker de antes:

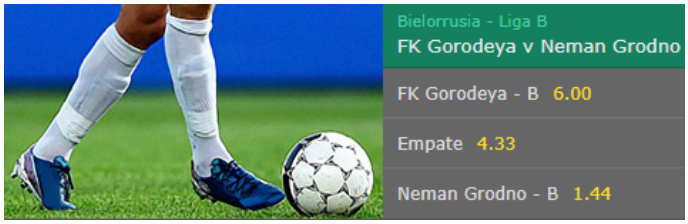


Figura 1.1: Ejemplo de cuota

Estas cuotas no son estáticas, varían en función de factores como, por ejemplo, el número de apuestas que se producen en torno a una de las opciones. Si se producen muchas apuestas en torno a una de las cuotas, y no se balancean las apuestas a la cuota contraria (u otras del mismo evento), veremos que las cuotas variarán minimizando la ganancia del usuario que viene a apostar a continuación a esa misma opción. En todo caso una vez apostado, en términos de la apuesta realizada, las cuotas se fijan en el momento de la confirmación de la apuesta y no cambian en caso de fueren mas o menos beneficiosas para el apostante.

Este fenómeno lo vemos cuando un tipster (los veremos mas adelante) famoso da a conocer una predicción. Entonces la gente que lo sigue comienza a apostar lo que el tipster ha pronosticado, generando que las cuotas bajen y den menos beneficios a los apostantes que acudan mas tarde a generar la apuesta.

Ilustramos a continuación un caso en Tenis profesional, un tipster (TipsterNavas en este caso) anuncia un pronostico vía Telegram donde pronostica que ganará Tabur, y la cuota que tenía él en el momento de la apuesta. Publicada a las 9:05 vemos que 1 hora y media después, vista la apuesta ya 1477 personas, la cuota ha bajado a 1.36 si lo comprobamos en el broker que el nos recomienda (Bet365):

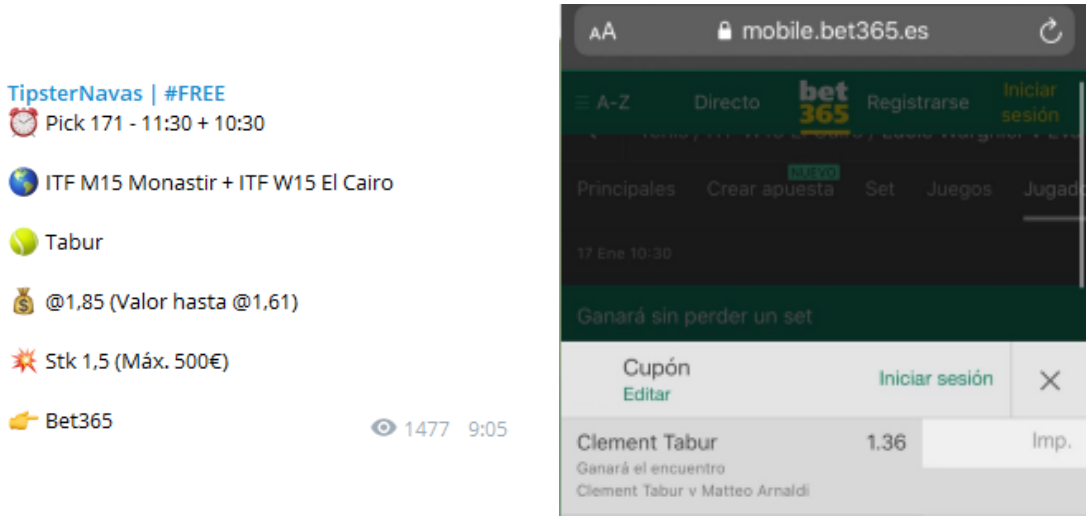


Figura 1.2: Cuotas llegando tarde al Pick.

Es un sistema por el que las casas de apuestas se blindan las espaldas. Estos brokers llevan muchas apuestas a la vez y sus sistemas podrían haber detectado mal una cuota inicial. En base a la experiencia y conocimiento que poseen los tipsters, estos últimos ven una oportunidad si la cuota esta mas alta de lo que consideren justo. Fijándonos en el ejemplo anterior, TipsterNavas no recomienda apostar si la cuota baja de @1,61.

Estar tan pendiente de ser de los primeros que sigue la apuesta de un tipster es engorroso, esto es uno de los motivos por los que hacemos una app que no sería tan conocida, para poder adelantarnos al resto y obtener mejores cuotas.

Nosotros nos vamos a acotar, quedando dentro del mercado de de **La Liga** española. Dejaremos aún así la aplicación preparada para que fuese fácil añadir otras competiciones, eso si, siempre fútbol.

Para las apuestas hemos decidido que vamos a elegir las siguientes:

- **Goles**
- **Córners**
- **Tarjetas**

El motivo principal de la elección de estas apuestas en concreto, es que creemos que son las que pueden seguir un modelo estadístico predictivo [3] para que nuestras predicciones superen la precisión de las del broker, pudiendo obtener así mejor cuota de apuesta.

Un ejemplo de uso de nuestra aplicación, sería comprobar el pronóstico para tarjetas en el partido, y después, la sección de tarjetas que ofrece el broker para ese evento. Si nuestro pronóstico fuese por ejemplo 6.23 tarjetas en el encuentro, y nos encontramos algo como esto:

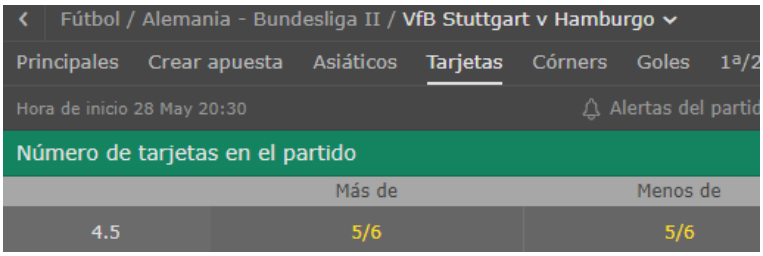


Figura 1.3: Ejemplo de apuesta de tarjetas

El broker pronostica que habrá en torno a 4.5, si nuestra app nos da un numero notablemente mayor, es recomendable apostar a que habrá mas de 4.5

Si nuestro pronóstico y el del broker son ajustados, lo cual ocurre en una cantidad elevada de casos, es una apuesta que se acerca al 50% de probabilidad de salir beneficiosa, así que entran en cuenta las ganas de jugársela el apostante.

Motivamos nuestro TFG debido a que las aplicaciones para pronósticos que encontramos son de pago en las categorías que nos interesan , casi todas ofrecen algún tipo de modo de prueba pero está bastante limitado, con el objetivo de poder captar al cliente y su dinero recurrente en forma de subscripción.

El sistema que implementaremos se centrará en llegar a la gente fácilmente, por lo que será lo mas conciso, breve e intuitivo posible para el usuario.

Queremos que sea muy sencillo, que muestre los próximos partidos de La Liga, que puedas entrar en ellos y obtener rápidamente la información, y proporcionarle al usuario que tenga curiosidad algunos datos de enfrentamientos previos.

Para llevar todo esto a cabo, marcamos 3 objetivos primordiales.

- Creación de una ETL y un modelo de datos coherente a nuestras necesidades.
- Creación de un servicio REST que exporte los datos que requerirán las funcionalidades de la app.
- Creación de una aplicación móvil para mostrar las predicciones al usuario.

Para que el usuario tenga una visión mas amable de la predicción, a modo de histórico, le mostraremos los resultados anteriores entre los dos equipos que involucran el partido que se pronostica.

En el Capítulo 2 hablaremos del estado actual del mercado de pronósticos deportivos mas en detalle, después en el Capítulo 3 nos centraremos en explicar la metodología y las herramientas que necesitaremos para hacer realidad nuestra aplicación.

En el Capítulo 4 nos centraremos en evaluar la planificación del proyecto, qué recursos necesitaremos y qué costes asociados tendrá. En el Capítulo 5 estará el grueso del proyecto, ahí entraremos en materia explicando qué se hizo en las diferentes iteraciones y podremos ver cómo muta la idea inicial a medida que va tomando forma.

Finalmente en el Capítulo 6 relatamos lo aprendido con este TFG, en adelante estarán los apéndices, material adicional y la bibliografía.

Capítulo 2

Contextualización y estado del arte

A día de hoy existen multitud de sitios web que tienen una sección de pronósticos donde te asesoran sobre las tendencias en las apuestas. Dichas páginas web están vinculadas a las casas de apuestas por sistemas de referido, en los enlaces que te ofrecen para seguir una apuesta, vemos un código de referencia para llevarse una parte del beneficio de la apuesta. A priori podría sonar a que las casas de apuestas están perjudicándose a si mismas permitiendo sitios que te asesoren sobre como apostar, pero nada mas lejos de la realidad. Ofreciendo estos servicios terminan siendo los que ofrecen el producto (Apuestas) y al mismo tiempo son los que tienen control sobre como te asesoran sobre él, pudiendo guiar al consumidor para conseguir una mayor confianza a la hora de apostar, que termina convirtiéndose en un beneficio para la casa.

Las casas son las que determinan las cuotas sobre una apuesta, y al poder gestionarla ellos mismos, mediante algoritmos basados en el número de personas que apuestan, pueden cambiar las cuotas rápidamente a su favor. Generalmente el método de control de cuotas es calculado por algoritmia en base a un análisis de resultados previos siguiendo un modelo matemático como nos explican en Betsson (Casa de apuestas) [4], lo que también incluye un cálculo en caliente de cuotas como explicamos en el Capítulo 1.

Otro punto de apoyo que tienen las casas de apuestas son los programas de televisión de ayuda o consejos en quinielas, etc. A priori parecen tertulias desinteresadas pero podemos ver que las casas de apuestas ocupan un puesto privilegiado en las cadenas de televisión, pagando para figurar promocionadas en anuncios y en espacios exclusivos.

Este hecho fomenta el problema de la ludopatía, gente adicta a apostar y que, lamentablemente, pierde la razón y la capacidad de controlar el gasto que le supone estar apostando. Terminan cayendo en un círculo vicioso de ganar-y-perder, finalizando generalmente con una pérdida que suele afectar también a temas fuera del ámbito económico.

Podríamos pensar que este problema esta vinculado a la edad adulta, pero nada mas lejos de la realidad. Los anuncios de casas de apuestas se producen durante las noticias e incluso entre pausas en los programas dirigidos a los mas jóvenes. Al ser un ambiente de horario para todos los públicos, comenzamos a ver a menores que se cuelan y frecuentan casas de apuestas.

En la siguiente gráfica vemos las preferencias de apuesta segregando por edades jóvenes y avanzadas. Se percibe que el juego online (nuestro ámbito de estudio) es la preferida para las edades jóvenes.

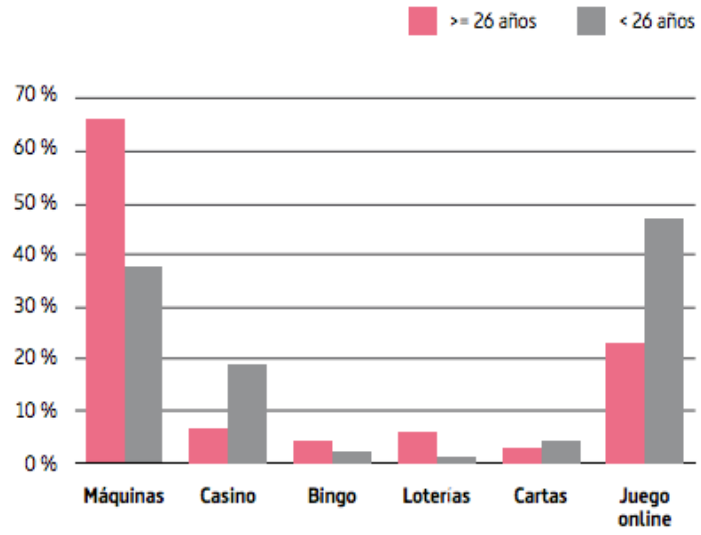


Figura 2.1: Estudio de FEJAR. Artículo de la Universidad Francisco de Vitoria [1]

Recientemente el Gobierno Español ha querido tomar cartas en el asunto regulando la publicidad de las casas de apuestas durante estos espacios televisivos, en concreto cuando el evento esta dirigido a los menores de edad [5]. Esto no es nada nuevo, en Italia los han regulado con la mano más dura posible, actualmente están prohibidos incluso durante los propios eventos deportivos [6]. Como dato, el Real Madrid tuvo de patrocinador a BetFair durante los años 2007-2013 [7] y el FC Barcelona recientemente cambió a Betfair por 1XBet al terminar el contrato de 3 años que firmaron en 2019. [8]

El mundo de la web en cambio se deja de lado, es debido a que se percibe que un menor tendrá más difícil toparse con estas páginas si no las busca previamente. Las diferentes páginas que aparentemente están desvinculadas de las casas de apuestas y que ofrecen servicios de consejos sobre apuestas, “Tips” de ahora en adelante, están ligadas a dichas casas de apuesta mediante sistemas de referido o invitaciones, llevándose una comisión por cada cliente que les brindan.

Ilustramos con el caso de una página así, OddsChecker. Dentro de la página elegimos un pronóstico aleatorio, el primero que nos brindan por ejemplo:

El peor local recibe al peor visitante

- Aun en descenso, el Werder ha firmado un **siete de nueve que aviva sus esperanzas** de salvación; el Eintracht, un **cuatro de seis que le aleja de la quema**.
- Los blanquiverdes son el **peor equipo de la Bundesliga en casa (seis puntos en 13** partidos); las "águilas", el **peor a domicilio con 10 en 14**.
- Los locales han basado su mejoría en torno al cerrojo de su portería: **imbatidos tres jornadas seguidas** y un global de solo **dos goles en tres partidos**.

Enfrentamientos Previos (Últimos 6 partidos)

	Ganados (33%)	Empatados (33%)	Ganados (33%)	
9 11 -	2/6	2/6	2/6	10 7 2

GDP	04-03-2020	F	1 **Eintracht Frankfurt** 2 : 0	Werder Bremen	
GB	06-10-2019	F	Eintracht Frankfurt 2 : 2	Werder Bremen	
GB	26-01-2019	F	Werder Bremen 2 : 2	Eintracht Frankfurt	
GB	01-09-2018	F	1 Eintracht Frankfurt 1 : 2	**Werder Bremen**	
GB	01-04-2018	F	**Werder Bremen** 2 : 1	Eintracht Frankfurt	
GB	03-11-2017	F	**Eintracht Frankfurt** 2 : 1	Werder Bremen	

Pick: El Werder Bremen gana o empata y menos de 3,5 goles. bet365

El conjunto *werderaner* estaba desahuciado hace pocas semanas. No obstante, los de Kohfeldt han lavado drásticamente su imagen defensiva y saldrían del descenso si ganan su partido atrasado ante el Eintracht, curiosamente el peor visitante del campeonato. Eso sí, la tendencia global nos obliga a amarrar un poco más con la doble oportunidad local y un *under* elevado.

Figura 2.2: Pronóstico deportivo en OddsChecker.

Vemos cómo nos comentan el partido, nos enseñan algún dato previo para hacernos entender que son expertos, que lo han mirado bien, todo con el objetivo de ganar nuestra confianza. Finalmente te muestran el Pick (La apuesta que proponen en cuestión) que recomiendan y qué les motiva a pensarlo. Si lo analizamos bien prácticamente están dando una opinión, no encontramos nada realmente sólido sobre lo que apoyar la realidad del pronóstico más que la experiencia que aseguran tener, entonces estamos entrando en una mecánica de confianza ciega. Te dan un enlace al broker para que puedas llevarla a cabo cómodamente, en el enlace esta contenido su código de referido.

Nosotros iremos más lejos, queremos tener un motivo matemático para decir el por qué de nuestro pronóstico, es debido a eso que elegimos minuciosamente los tipos de apuestas en los cuales centrarnos (Goles, tarjetas y corners), dado que creemos que son los más sencillos para poder justificar matemáticamente.

Cómo vimos antes, si analizamos los enlaces que nos brindan para realizar tu apuesta, vemos que son redirecciones a la casa de apuestas donde son afiliados, con su enlace de socio complementando la URL del enlace que te ofrecen. Estas páginas se basan tanto en la matemática como en la experiencia de personas conocedores del medio. Son un sistema híbrido entre matemática y experiencia de sus asesores.

Otra modalidad muy extendida es confiarle tus apuestas a una persona en singular en lugar de a una casa. Dicha persona es, a efectos de percepción, un gurú en la materia. Estas personas son los llamados Tipsters, gente que anuncian resultados de apuestas mediante redes sociales, y que ofrecen un servicio de venta de información privada y/o privilegiada para mejorar los resultados de las apuestas de sus clientes. basándose mas en su opinión y experiencia que en los datos directamente.

También vemos la existencia de apps donde se engloban varios tipsters a la vez dando sus pronósticos a cambio de una subscripción que suele ser mensual.

No es raro que existan estas apps. Es un nicho de mercado crear un marketplace donde los tipsters puedan publicar sus pronósticos y tener gente subscrita que se beneficia de ellos.

Vemos el caso de Apptipsters.com:

Figura 2.3: Aplicación AppTipsters

Esta aplicación permite exactamente eso:

- Los tipsters se dan de alta
- Pueden publicar sus picks
- Eligen los costes para seguirlos

Finalmente los usuarios que los sigan proporcionarán un método de pago y se les cobra. El diseñador de la app se lleva una parte y supuestamente todo el mundo gana.

Generalmente los Tipsters se dan a conocer en perfiles de redes sociales. Tanto por el alcance que tienen como porque son gratis para el usuario captado y para el Tipster. Analizamos el caso de Apuestas Deportivas (Twitter). Lo primero que nos encontramos es que directamente ofrece una app alternativa para seguirle:

Figura 2.4: Tweet anclado del Tipster Apuestas Deportivas.

Los Picks de los Tipsters no dejan de ser opiniones personales de alguien con experiencia en materia, nuevamente, no nos convence algo que no tenga un soporte demostrable lógicamente.

Son muy comunes los Tipsters que pillan una buena racha, ganan muchos subscriptores, y luego fallan estrepitosamente en sus siguientes predicciones.

Véase el caso del Tipster kenesisa69, experto en fútbol:

Historial | Momento | Stake | Cuota | Bookie | Mes actual

Mes	Apuestas	Beneficio (uds)	Yield	Cuota media	Stake medio	Win rate
Jun 2020	1	-1.50	-100.00%	1.72	1.50	0.00%
May 2020	9	-1.50	-10.71%	1.85	1.56	44.44%
Mar 2020	15	-3.56	-12.71%	2.17	1.87	40.00%
Feb 2020	66	+14.15	11.94%	1.99	1.80	59.09%
Ene 2020	54	-11.95	-13.13%	2.46	1.69	37.04%
Dic 2019	34	+29.26	50.02%	2.14	1.72	70.59%
Nov 2019	59	+20.92	19.88%	2.11	1.78	50.85%
Oct 2019	85	-2.23	-1.36%	2.14	1.92	44.71%
Sep 2019	65	-25.82	-21.61%	2.14	1.84	35.38%
Ago 2019	5	+0.80	7.62%	1.97	2.10	60.00%
Jun 2019	62	+15.79	14.69%	2.11	1.73	54.84%
May 2019	59	+40.96	40.35%	2.29	1.72	66.10%
Abr 2019	52	+22.07	22.99%	2.07	1.85	59.62%
Mar 2019	67	+13.15	10.40%	2.02	1.89	53.73%
Feb 2019	75	+6.82	5.31%	2.32	1.71	49.33%
Ene 2019	55	+32.09	33.08%	2.11	1.76	61.82%

Figura 2.5: Historial del Tipster Kenesisa69.

Empieza acertando muchas predicciones, su número de seguidores aumenta, y finalmente está encadenando meses de pérdidas.

Nosotros nos vamos a ubicar en la zona mas empírica posible, ciñendo nuestros pronósticos a la observación y al análisis del dato. Recopilaremos datos de enfrentamientos previos de los equipos, y sólo tomaremos decisiones en base a esos datos.

Capítulo 3

Metodología y Herramientas

3.1 Metodología

Utilizaremos un modelo iterativo e incremental. Este modelo establece que el proyecto se planifica en distintos bloques temporales a los que nos referiremos como iteraciones. En cada iteración se van añadiendo funcionalidades nuevas, hasta la versión final no estarán todas, pero las que estén estarán completas.

Para conseguir esto, cada requisito debe tener un completo desarrollo en una única iteración debiendo quedar bien documentado e incluir pruebas.

Aplicado al software, la metodología incremental nos permite ir evolucionando el producto desde el primer momento de su concepción. En cada iteración se van añadiendo nuevas funcionalidades hasta que finalmente el producto esta completo. Es importante hacer partícipe de esto al usuario para que le sea más cómoda la aceptación del producto, en otras palabras, que sea adecuado a la idea mental que él tenía originalmente.

En esencia, este modelo se basa en dos premisas:

- Divide y vencerás. Separando las funcionalidades en bloques más simples se hacen más fáciles de diseñar, y el acoplamiento entre partes es menor.
- Durante el desarrollo, los procesos tienden a cambiar, en una iteración posterior pueden detectarse problemas de una iteración pasada con las pruebas de integración.

En la metodología incremental iterativa segmentamos en macro requisitos el producto final y vamos completando iteraciones donde solventamos en cada una diversas funcionalidades pasando por unas fases de análisis, diseño, implementación y pruebas.

Existe un análisis inicial a un nivel muy elevado que nos permite establecer los macro requisitos y dividir qué tiene cabida en cada iteración. Esto es una fase inicial que, aunque sea considerada una iteración, carece de diseño, implementación y pruebas.

Explicamos a continuación cada una de las 4 fases de una iteración normal:

3.1.1 Análisis

En esta fase ubicamos el trabajo que nos requiere conocer el alcance del objetivo de la iteración. Se analiza lo que se quiere conseguir y de lo que partimos de base.

Lo mas importante en el análisis es tener claro cuales son los objetivos a cumplir para la iteración en cuestión. Es importante saber en concreto qué funcionalidades del producto final se realizarán y qué necesidades tenemos pero sin entrar en diseño, en el análisis es donde tenemos que calcular o al menos aproximar los costes de esa iteración.

En cada iteración se añaden funcionalidades nuevas. Al final de cada iteración el resultado es un producto con funcionalidades completas. No figurarán todas hasta la versión final, pero las que figuren estarán completas.

3.1.2 Diseño

Tras la fase de análisis, donde respondemos al 'qué', el diseño representa el 'cómo'.

En esta fase tenemos que establecer que patrones de diseño encajan mejor con nuestros objetivos. Nos debemos centrar en ver cómo vamos a hacer para implementar posteriormente las funcionalidades que nos piden antes de ponernos expresamente a hacerlo.

En este paso estamos planeando como abordar las dificultades y metas que conocemos de la fase anterior.

3.1.3 Implementación

En la implementación se ubica todo el proceso de construcción que se ideó en el diseño. En el caso del software, sería toda la codificación y realización las especificaciones técnicas necesarias para alcanzar el objetivo.

3.1.4 Pruebas

Finalmente en esta fase debemos probar que todo lo implementado cumpla su función de forma adecuada. El software desarrollado pasará diversas pruebas unitarias y de integración.

Debe incluirse en las pruebas de integración la interacción con elementos desarrollados en iteraciones anteriores con los que tengan una dependencia funcional.

Además debemos establecer unos criterios de calidad para poder valorar nuestro producto. Generalmente estos criterios son que cumpla en costes razonables (tiempo y dinero) todos los requisitos funcionales que se pactasen al inicio del proyecto.

3.2 Metodología Aplicada a nuestro caso

En nuestro caso concreto, separamos nuestro proyecto en 3 macro requisitos y abordamos cada uno en su iteración pertinente como veremos en el Capítulo 5.

Nuestra aplicación final necesita datos de partidos de fútbol, en concreto, todo lo referente a goles, tarjetas y córners. Necesitamos extraer dichos datos de alguna fuente, esta fuente es lo que llamamos el proveedor externo. Es necesario tener una base de datos acorde para recibir la información y alguna forma de exportarlo a la red para que cualquier usuario de nuestra aplicación móvil pueda acceder y recuperar la información sin importar el lugar de conexión.

Lo abordamos de la siguiente forma:

1. Realizamos un análisis inicial, que llamaremos Iteración 0, para conocer el alcance y costes del proyecto, las herramientas software necesarias y elegir el proveedor de datos, también instalaremos todo lo necesario para poder comenzar la siguiente iteración con todas las herramientas conocidas y funcionales.

2. La Iteración 1 tiene como objetivo conseguirnos los datos necesarios para la aplicación. El desarrollo de una ETL completa y planificada que nos permita llevar la información desde el proveedor hasta nuestra propia BBDD, estableciendo una gestión de acceso a los datos que sea cómoda para nuestro servidor REST.

3. En la Iteración 2, crearemos el servidor REST, debemos definir unas funcionalidades a exportar y realizar la codificación y configuraciones de red pertinentes para ser alcanzables desde el exterior. La aplicación accederá desde direcciones externas que generalmente serán diferentes para cada usuario.

4. En la Iteración 3, nos centraremos en la aplicación móvil. Diseñaremos las diversas pantallas para poder dar los servicios que nos hemos comprometido a dar.

En el Capítulo 4 explicaremos con precisión la planificación de las iteraciones.

3.3 Herramientas

Utilizaremos las siguientes herramientas en sus versiones de código abierto para llevarlo a cabo:

- En la parte de ETL (Extracción, Transformación, Load (Carga)) de los datos, utilizaremos **Pentaho Community Edition 9.0 Data Integrator**[9, 10, 11] debido a que aunque tenemos conocimiento en herramientas mas potentes como **Informática PowerCenter**, esta última no tiene licencia gratuita.

- Para el almacenamiento y la administración de los datos: **MySQL 8.0.20 Database y MySQL Workbench** [12] por simplicidad, es muy cómodo tanto generar la base de datos y administrarla, para comunicarla con Pentaho, nos proporcionan también gratuitamente el driver JDBC apropiado.

- Para servir la información mediante un servicio web, utilizaremos: **Kotlin 1.3.72**[13] por conocimiento del lenguaje y las facilidades que ofrece para depurar un error. Dentro de Kotlin se utiliza **Java 8**[14] y **Spring**[15] (Utilizamos el website `https://start.spring.io/` para generar un proyecto Spring con las dependencias ya creadas).

- Para el desarrollo del frontal donde mostrarlo, utilizaremos: **Android Studio 3.6**[16] por conocimiento del mismo debido a asignaturas previas.

La version target será **Android 10 (API level 29)** y minima soportada **Android 6 (API level 23)**

Capítulo 4
Planificación

Para llevar a cabo en costes razonables el proyecto, utilizaremos un modelo iterativo, dedicando un importante tiempo inicial a conocer las tecnologías que nos son necesarias. Estableceremos la siguiente planificación dividida en fases que explicaremos a continuación.

4.1 Fases del proyecto:

Siguiendo el modelo que describimos en el capítulo anterior, existirá una iteración 0 donde solo tenemos fase de análisis y que tendrá los siguientes objetivos:

- Definir la arquitectura general del sistema.
- Adquirir el conocimiento sobre el software que necesitaremos en las siguientes iteraciones.
- Definir el proveedor de datos, calcular los costes del proyecto.

Hemos dividido el proyecto en 3 iteraciones porque consideramos que las 3 áreas diferenciadas son:

- Herramienta ETL y base de datos.
- Webservice API REST.
- Aplicación Móvil.

Condicionaremos el desarrollo y evolución de cada una de las áreas a las necesidades que surjan en las otras áreas.

Si detectamos algún problema sobre una de las iteraciones anteriores lo solventaríamos en la iteración donde lo hemos descubierto, eso podría ampliar los plazos de dicha iteración. Lo veremos con mas detalle en el seguimiento.

Mostramos de forma tabular en la siguiente figura las tareas asignadas a cada iteración y su duración. En la figura 4.2 lo mostraremos en forma de diagrama de Gantt

Tipo	Asunto	Fecha de inicio	Fecha de finalización
Phase	**Iteración 0**	**06/05/2020**	**14/05/2020**
Task	Definición de requisitos y arquitectura general del sistema.	06/05/2020	07/05/2020
Task	Ánalisis y selección del proveedor de datos	08/05/2020	08/05/2020
Task	Formación Pentaho	09/05/2020	10/05/2020
Task	Formación MySQL Workbench	11/05/2020	11/05/2020
Task	Formación Spring	12/05/2020	12/05/2020
Task	Formación Android Studio	13/05/2020	14/05/2020
Phase	**Iteración 1 ETL BBDD**	**15/05/2020**	**21/05/2020**
Task	Análisis	15/05/2020	16/05/2020
Task	Diseño	17/05/2020	17/05/2020
Task	Implementación	18/05/2020	20/05/2020
Task	Pruebas	21/05/2020	21/05/2020
Phase	**Iteración 2 API REST**	**22/05/2020**	**25/05/2020**
Task	Análisis	22/05/2020	22/05/2020
Task	Diseño	23/05/2020	23/05/2020
Task	Implementación	24/05/2020	25/05/2020
Task	Pruebas	25/05/2020	25/05/2020
Phase	**Iteración 3 - Aplicación Android**	**26/05/2020**	**01/06/2020**
Task	Análisis	26/05/2020	26/05/2020
Task	Diseño	27/05/2020	28/05/2020
Task	Implementación	29/05/2020	31/05/2020
Task	Pruebas	01/06/2020	01/06/2020

Figura 4.1: Planificación de iteraciones.

Explicamos brevemente las tareas dentro de cada área:

4.1.1 Iteración 0:

Definiremos la arquitectura general del sistema, seleccionaremos un proveedor de datos, decidiremos las herramientas que vamos a utilizar en las iteraciones siguientes y dedicaremos un tiempo amplio a formarnos en las herramientas software que necesitaremos.

Dada la naturaleza de nuestra aplicación, al tener que hacer predicciones debemos utilizar un modelo estadístico formal que nos permita conseguir un pronóstico acertado. Decidiremos dicho modelo en esta iteración.

4.1.2 Iteración 1: ETL BBDD

En la iteración 1, nos centraremos en desarrollar la base de datos y la carga ETL.

Analizaremos los datos que queremos conseguir, tenemos que dedicar tiempo a entender el modelo del proveedor para poder explotarlo con comodidad, y analizaremos también el sistema que planteamos para almacenar los datos.

Diseñaremos un modelo Entidad-Relación que alimentaremos con nuestra ETL, también diseñaremos unas vistas de acceso a los datos (vistas SQL ANSI-SPARC) donde definiremos la lógica matemática del modelo estadístico. Implementaremos nuestros diseños con un modelo relacional y finalmente haremos pruebas de unidad y integración para ver que la base de datos termina poblada y que que los datos son correctos.

4.1.3 Iteración 2: API REST

En esta iteración el objetivo es conseguir que los datos de nuestra base de datos sean accesibles desde la red de forma controlada.

Analizaremos qué endpoints vamos a publicar. Debemos configurar las redes para que fuésemos accesibles desde el exterior.

Diseñaremos nuestro servicio siguiendo patrones de diseño, dado que la función de esta API REST es solo exportar datos, ya adelantamos que vamos a utilizar DTO y en la aplicación móvil ya diseñaremos los DAO.

Implementaremos la API REST utilizando Kotlin y Spring. Haremos pruebas unitarias y de integración con JUnit y probaremos también llamando al endpoint desde diversas redes y navegadores. Debemos probar no solo que somos accesibles si no que los datos que devolvemos son correctos y contrastados con la base de datos.

4.1.4 Iteración 3 - Aplicación Android

Diseñaremos la aplicación en base a un patrón típico como Modelo-Vista-Controlador, que funciona muy bien en estas aplicaciones por pantallas. También es necesario realizar un estudio del flujo entre pantallas para que cumpla nuestra base, ser lo mas intuitiva posible.

Propondremos una serie de pruebas centradas en los cambios de pantalla, ver si se llaman bien los métodos deseados simulando la interacción del usuario.

A continuación añadimos el diagrama de Gantt del proyecto:

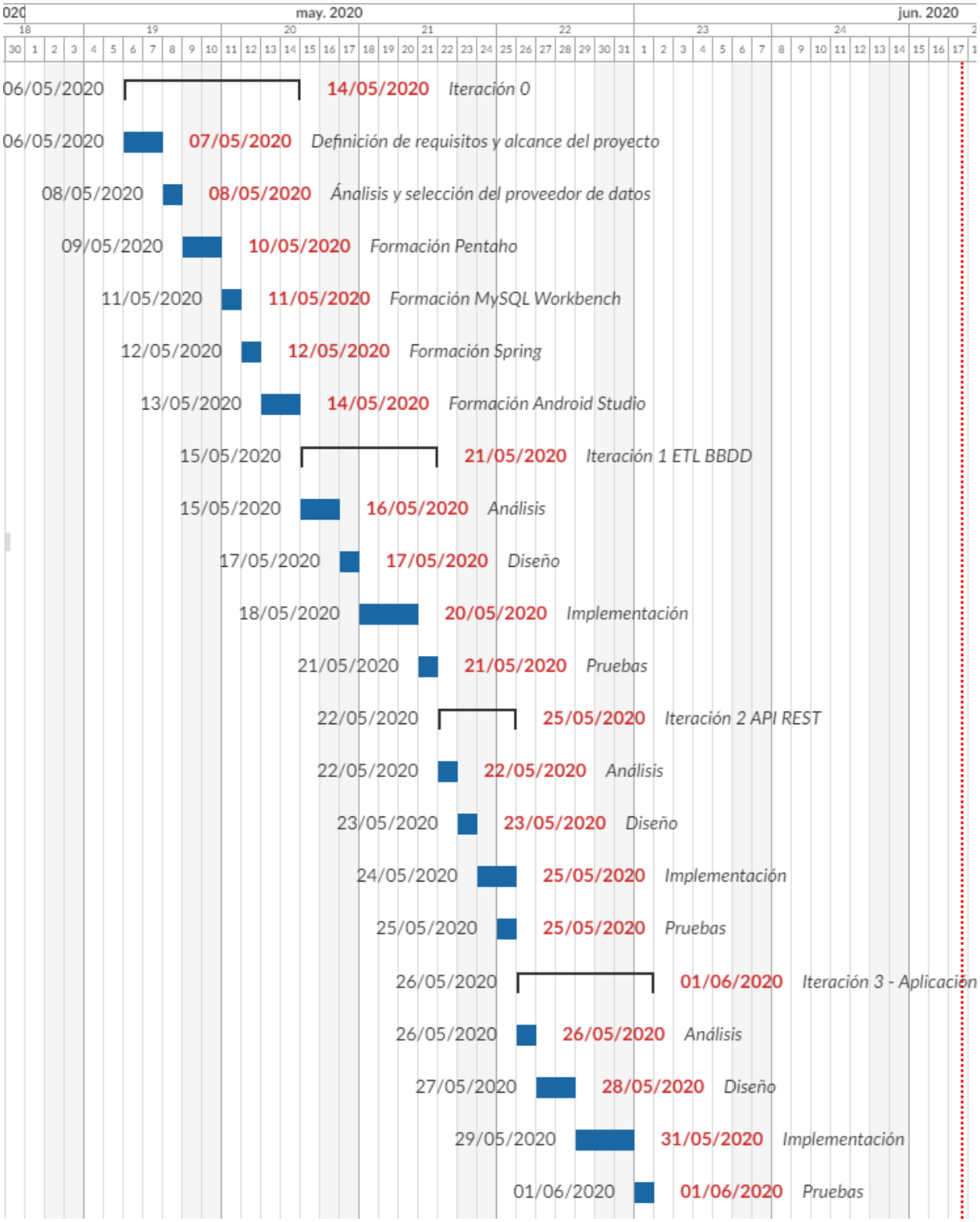


Figura 4.2: Diagrama de Gantt - Estimación.

4.2 Recursos

Utilizamos dos tipos de recursos para llevar a cabo el proyecto:
Recursos Personales

- **Analista**: Persona con capacidad de interpretación de los datos, debe saber recopilar datos a la vez que analizarlos de forma estadística
- **Programador**: Persona con conocimiento técnico para utilizar las herramientas software requeridas.

Recursos Hardware: (Por orden de flujo de datos)

- **Proveedor de datos**: Recurso externo, nos brinda la información en bruto.
- **Servidor de Datos**: Gestiona la herramienta ETL, la lógica de negocio, y el almacenamiento de nuestro dato.
- **Servidor REST**: Gestiona las peticiones de la aplicación Android, para brindarle la información necesaria desde nuestro origen.

4.3 Costes

Recurso	Unidades	Precio x Unidad	Total
Analista	40 horas	30€	1200€
Desarrollador	80 horas	22€	1760€
Servidor de Datos	1	575€	575€
Servidor REST	1	35€	35€
Proveedor de datos	3x Subscripción mensual	300	900

Figura 4.3: Costes de los recursos.

Figurando un coste total de proyecto de : **4470€**

4.4 Seguimiento del proyecto

Mostramos la siguiente tabla donde complementamos la figura 4.1 con la información de seguimiento del proyecto.

Tipo	Asunto	Inicio Estimado	Finalización Estimada	Inicio Real	Finalización Real	Retraso o adelanto
Phase	Iteración 0	06/05/2020	14/05/2020	**06/05/2020**	**14/05/2020**	0
Task	Definición de requisitos y alcance del proyecto	06/05/2020	07/05/2020	06/05/2020	07/05/2020	0
Task	Ánalisis y selección del proveedor de datos	08/05/2020	08/05/2020	08/05/2020	08/05/2020	0
Task	Formación Pentaho	09/05/2020	10/05/2020	09/05/2020	10/05/2020	0
Phase	Iteración 1 ETL BBDD	15/05/2020	21/05/2020	**15/05/2020**	**24/05/2020**	3
Task	Formación MySQL Workbench	11/05/2020	11/05/2020	11/05/2020	11/05/2020	0
Task	Formación Spring	12/05/2020	12/05/2020	12/05/2020	12/05/2020	0
Task	Formación Android Studio	13/05/2020	14/05/2020	13/05/2020	14/05/2020	0
Task	Análisis	15/05/2020	16/05/2020	15/05/2020	16/05/2020	0
Task	Diseño	17/05/2020	17/05/2020	**17/05/2020**	**19/05/2020**	2
Task	Implementación	18/05/2020	20/05/2020	**20/05/2020**	**23/05/2020**	1
Task	Pruebas	21/05/2020	21/05/2020	23/05/2020	23/05/2020	0
Phase	Iteración 2 API REST	22/05/2020	25/05/2020	**25/05/2020**	**27/05/2020**	-1
Task	Análisis	22/05/2020	22/05/2020	25/05/2020	25/05/2020	0
Task	Diseño	23/05/2020	23/05/2020	26/05/2020	26/05/2020	0
Task	Implementación	24/05/2020	25/05/2020	**27/05/2020**	**27/05/2020**	-1
Task	Pruebas	25/05/2020	25/05/2020	27/05/2020	27/05/2020	0
Phase	Iteración 3 - Aplicación Android	26/05/2020	01/06/2020	**28/05/2020**	**06/06/2020**	3
Task	Análisis	26/05/2020	26/05/2020	28/05/2020	28/05/2020	0
Task	Diseño	27/05/2020	28/05/2020	**29/05/2020**	**31/05/2020**	1
Task	Implementación	29/05/2020	31/05/2020	**01/06/2020**	**05/06/2020**	2
Task	Pruebas	01/06/2020	01/06/2020	06/06/2020	06/06/2020	0

Figura 4.4: Seguimiento del proyecto.

Surgieron problemas en el diseño ETL debido a que hemos tenido que solventar que el proveedor nos proporcione los partidos de forma paginada. Nos ha llevado 2 días extra conseguir una solución al problema. Esto mismo se ve reflejado en la implementación, originalmente habíamos pensado una carga más básica pero ha llevado más tiempo debido a tener que implementar una compleja solución por bucles.

En la segunda iteración hemos ganado un día de trabajo porque la implementación en Kotlin de una API REST usando Spring es muy sencilla. Basándonos en nuestras anteriores experiencias trabajando con servidores REST pensamos que iba a ser más laborioso.

En la aplicación Android perdemos 3 días porque tuvimos problemas para mostrar la información meteorológica de los partidos. Tardamos bastante en encontrar el error que lo producía. Además las pantallas resultaron mas costosas de lo esperado.

Capítulo 5

Desarrollo del Proyecto

En este capítulo hablaremos del núcleo del proyecto, detallando cada iteración con ejemplos y concretando los retos que nos surgieron.

5.1 Iteración 0

Las tecnologías que necesitaremos son las descritas en el apartado 3.3. Los costes se especificaron con anterioridad también en el apartado 4.3.

En esta fase del proyecto, donde analizamos su alcance, nos toma un tiempo importante comprender la estructura del proveedor de datos. En base a ella, diseñaremos el modelo de datos y la apariencia inicial que queremos para nuestra app.

El proveedor de datos escogido es : SportMonks.com

Fue elegido como proveedor porque aporta realmente mucha información de cada partido y tiene una API bien documentada [17] que nos sirvió para encontrar rápidamente lo que necesitábamos.

Ellos distribuyen la información de diversas formas, pero la única que nos presentaba una relación completa entre Goles-Tarjetas-Corners para equipo visitante y local, sería descargarnos la información a nivel de partido y por rango temporal, y después filtrar por temporadas.

Explicamos a continuación uno de los pilares de nuestro proyecto, el modelo estadístico que sustenta nuestras predicciones.

5.1.1 Arquitectura general.

La arquitectura general de nuestra aplicación es que los datos salen del proveedor en las cargas planificadas, se almacenan en nuestra base de datos, accedemos a ellos desde la API REST y los mostramos en la aplicación móvil.

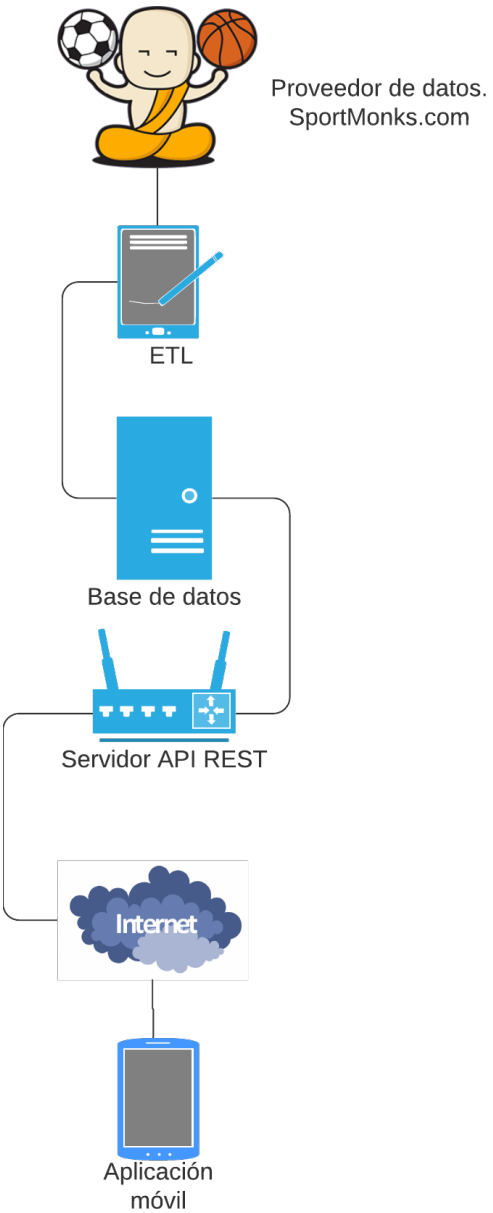


Figura 5.1: Diseño general de la aplicación

5.1.2 Modelo Estadístico.

Para la descripción matemática del modelo utilizaremos la distribución de Poisson [18]. La implementaremos en la zona de la BD para evitar incluir lógica en la API REST o en la aplicación móvil.

$$\boxed{P\left(x=k\right)=\frac{e^{-\lambda}\lambda^{k}}{k!}} \tag{5.1}$$

La distribución de Poisson es una distribución discreta de probabilidad que nos dice cual es la probabilidad de que un evento ocurra un numero K de veces dado que en promedio ocurre un numero λ de veces. Originalmente se usó para modelar el número de personas que van injustamente a la cárcel.

Podemos usar este modelo estadístico para estimar la probabilidad de eventos específicos.

Los factores del modelo, o factores relevantes, son el número de goles que anota un equipo y también el número de goles que recibe. Nosotros consideramos que un equipo que ha recibido muchos goles tiene una probabilidad alta de recibir goles en un futuro, de igual forma, un equipo que marca muchos goles es muy probable que marque muchos goles en un futuro.

En nuestro caso lo aplicaremos a los goles y córners. Nos centraremos en explicarlo para los goles pero se aplica de igual manera para los córners.

El número de goles marcados por cada equipo es considerado un evento independiente. En nuestra distribución de Poisson, el parámetro λ es el número promedio de goles que anota cada equipo. El número k es la probabilidad que queremos averiguar.

La primera parte es tener cuantificados todos los goles anotados por locales y visitantes y los goles encajados por locales y visitantes. Una vez conseguidos, se suman todos los goles en las categorías L/V y se calcula el promedio teniendo en cuenta el número de partidos.

Teniendo estos datos ya nos vamos al partido en cuestión que queremos pronosticar. Calculamos el numero de goles anotados y encajados para ese equipo en concreto. Con estos valores definimos dos valores nuevos, “Fuerza de ataque” y “Fuerza de defensa”.

La fuerza de ataque de un equipo es un promedio entre lo que anotan los equipos locales o visitantes entre lo que anota este equipo en concreto jugando en esa posición (local/visitante).

Cada equipo tiene 4 fuerzas. Su Ataque y Defensa como local y sus homólogos como visitante.

A continuación mostramos la ecuación de la fuerza de ataque de un equipo como local, para calcular las otras 3 fuerzas bastaría con cambiar los valores por los correspondientes cuando es visitante o cuando son goles encajados.

$$F.\ Ataque\ Local_{Equipo} = \frac{Goles\ Anotados\ Local_{Equipo}/Partidos\ Local_{Equipo}}{Goles\ Anotados\ Todos\ Local/Partidos\ Todos} \tag{5.2}$$

$$F.\ Defensa\ Local_{Equipo} = \frac{Goles\ Encajados\ Local_{Equipo}/Partidos\ Local_{Equipo}}{Goles\ Encajados\ Todos\ Local_{/}Partidos\ Todos} \tag{5.3}$$

Explicamos los términos empleados a continuación.

Término	Descripción
Goles Anotados Local	Sumatorio de todos los goles que un equipo ha anotado como local.
Goles Encajados Local	Sumatorio de todos los goles que un equipo ha recibido como local.
Partidos Local	Número total de partidos que se conocen del equipo jugando como local.
Goles Anotados Todos Local	Sumatorio de los goles de todos los equipos jugando como locales.
Goles Encajados Todos Local	Sumatorio de los goles que han recibido todos los equipos jugando como locales.
Partidos Todos	Número de partidos totales.

Figura 5.2: Tabla de términos.

Para conseguir los datos del visitante se aplicaría el mismo cálculo pero utilizando los valores propios de jugar como visitante.

En los sumatorios de todos los equipos el valor de “Goles Anotados Todos Local” coincidiría con “Goles Encajados Todos Visitante”, análogamente el valor de “Goles Encajados Todos Local” coincide con “Goles Anotados Todos Visitante”.

Para la "Fuerza de ataque" se tienen en cuenta los goles anotados, y para la "Fuerza de defensa" se tienen en cuenta los goles encajados.

Teniendo estos dos valores calculados, podemos calcular los goles esperados. Este valor esperado es la λ de nuestra distribución de Poisson.

La obtenemos multiplicando las fuerzas opuestas, es decir, los "Goles esperados" (λ) del local es su fuerza de ataque multiplicada por la fuerza de defensa del visitante y por el promedio de goles anotados por los locales en la competición. Y para el visitante es el cálculo recíproco.

$$Goles\ Esperados\ Local_{Eq1} = F.At.\ Local_{Eq1} * F.Def.\ Visitante_{Eq2} * Media\ Goles_{\ Todos\ los\ Locales} \tag{5.4}$$

Sumando los goles esperados del Local y del Visitante. Obtenemos la predicción a línea de gol que estamos buscando.

Iterando por el parámetro "k" de la distribución de Poisson, tendríamos la probabilidad de esa "k" cantidad de goles en concreto.

De cara a la aplicación móvil, realizamos un primer diseño de mockups de cómo sería la interfaz de usuario:

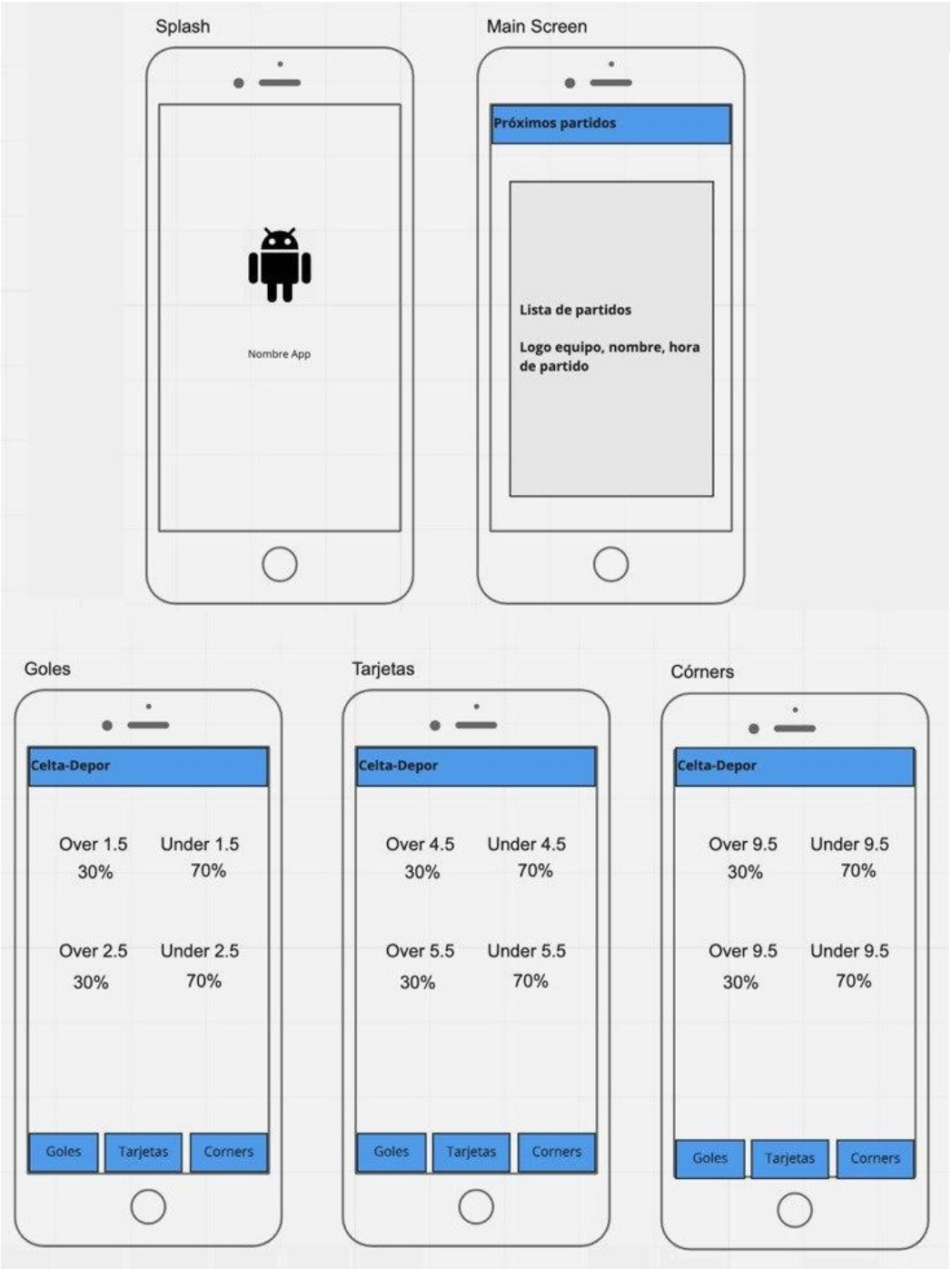


Figura 5.3: Mockups. Aplicación móvil

De las 3 iteraciones planificadas, la más breve creemos que será la de la API REST. Kotlin (con la ayuda de Spring) permite diseñar un servicio de este tipo de forma sencilla con prácticamente rellenar 2 ficheros: un primer fichero, especificando el tipo de los datos, y otro con los mappings de los endpoints y las consultas a lanzar.

La aplicación móvil la diseñaremos en Android. La idea es poder mostrar los diferentes partidos para cada jornada, y las previsiones a "Linea de Gol" (número de goles), tarjetas y córners. Son tres apuestas muy comunes en las casas de apuestas a día de hoy.

5.2 Iteración 1 - BBDD y ETL

En esta iteración buscamos completar la parte mas extensa del sistema, es decir, el diseño de la ETL y nuestra base de datos.

5.2.1 Análisis:

Analizaremos primero el modelo del proveedor para ubicar correctamente todos los datos que nos interesan y al mismo tiempo familiarizarnos con su modelo.

Tenemos claros los datos que queremos: los goles, las tarjetas y los córners de los partidos correspondientes a las 3 últimas temporadas vigentes de La Liga.

El proveedor establece en su documentación [17] que los datos de partido se solicitan por rango temporal, utilizando 2 fechas como parámetros.

Dado que los datos pasados son históricos y no deberían cambiar, la carga incremental solo la realizaríamos para para los últimos 30 días, recopilando así la última jornada y unas pocas mas, de cara a prever que pudiese variar algún dato en el servidor, por estar mal o por realmente haber mutado por una sanción, etc.

Así mismo, se establece la carga completa para las dimensiones necesarias, dado que tienen pocos registros.

A continuación mostramos la información de un partido concreto para que nos hagamos una idea del modelo de datos que maneja el proveedor. Hemos omitido los objetos que no nos aportan valor y, de los que sí, hemos resaltado en rojo los campos que nos interesan y que trabajará la ETL:

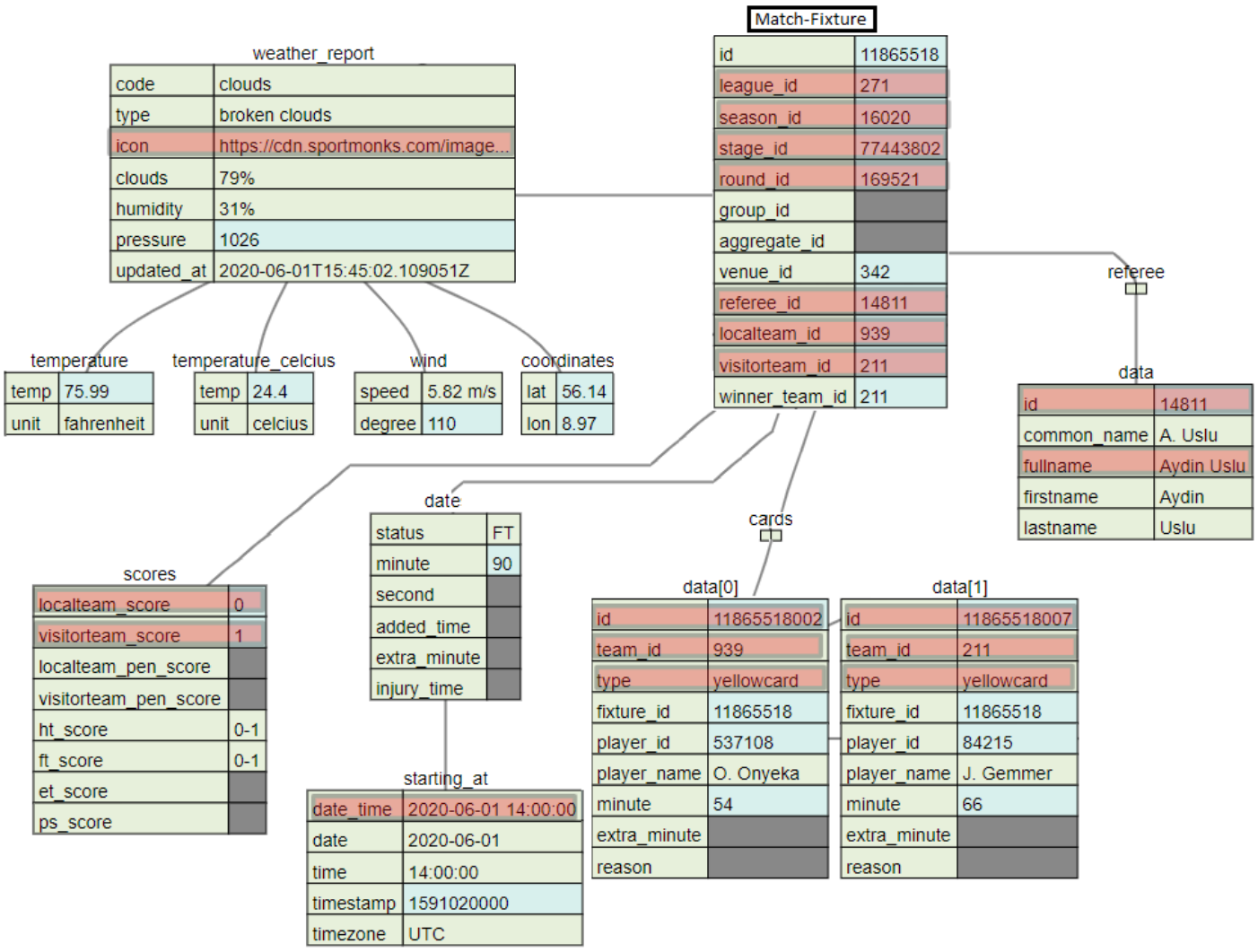


Figura 5.4: Ejemplo de datos de un partido concreto en el proveedor.

Tras analizar los datos en el servidor y viendo la necesidad de conocer para cada equipo sus estadísticas, tanto Local como Visitante, creemos que tenemos que tener la información agrupada a nivel de partido en nuestra base de datos.

Se fija entonces que los datos a recuperar son los siguientes:

- Últimas 3 temporadas de la liga.
- Jornadas asociadas a dichas temporadas.
- Todos los equipos participantes.
- Árbitros participantes.
- Todos los partidos de las últimas 3 temporadas de la liga.

Muy posiblemente necesitemos un área de Staging en la base de datos donde almacenar toda la información para trabajarla y ajustarla a nuestra necesidad.

La información propia de los equipos, el árbitro, la temporada y la jornada; la tendremos almacenada en sus respectivas dimensiones.

Nos encontramos el problema de que el proveedor no tiene un acceso directo a los datos de los árbitros, así que tenemos que recuperarlo de los partidos y construir la dimensión en base a ello.

Buscamos tener una base de datos que tenga directamente los datos finales y que prácticamente no tenga que aplicarse lógica en la capa de aplicación, por lo que vamos a necesitar realizar una ETL que refine bien los datos y además vamos a tener que crear unas vistas de datos donde implementaremos el modelo matemático para que la aplicación muestre directamente lo que recupera.

Se desea que el trabajo que realice nuestra API REST sea prácticamente recuperar los datos procesados, sin tener que aplicar lógica en su capa. Por lo que en las vistas que codifiquemos deben ir los valores del pronóstico.

Los datos que queremos ofrecer se agrupan en torno a los partidos. Debemos diseñar un modelo ER que tenga como centro una tabla donde agrupemos eses valores totalmente calculado (lo haremos en la ETL) a nivel de un único registro por partido.

Partiendo del modelo del proveedor (Figura 5.4), diseñamos un modelo ER que nos permita almacenar todos los datos relevantes para nuestro aplicativo.

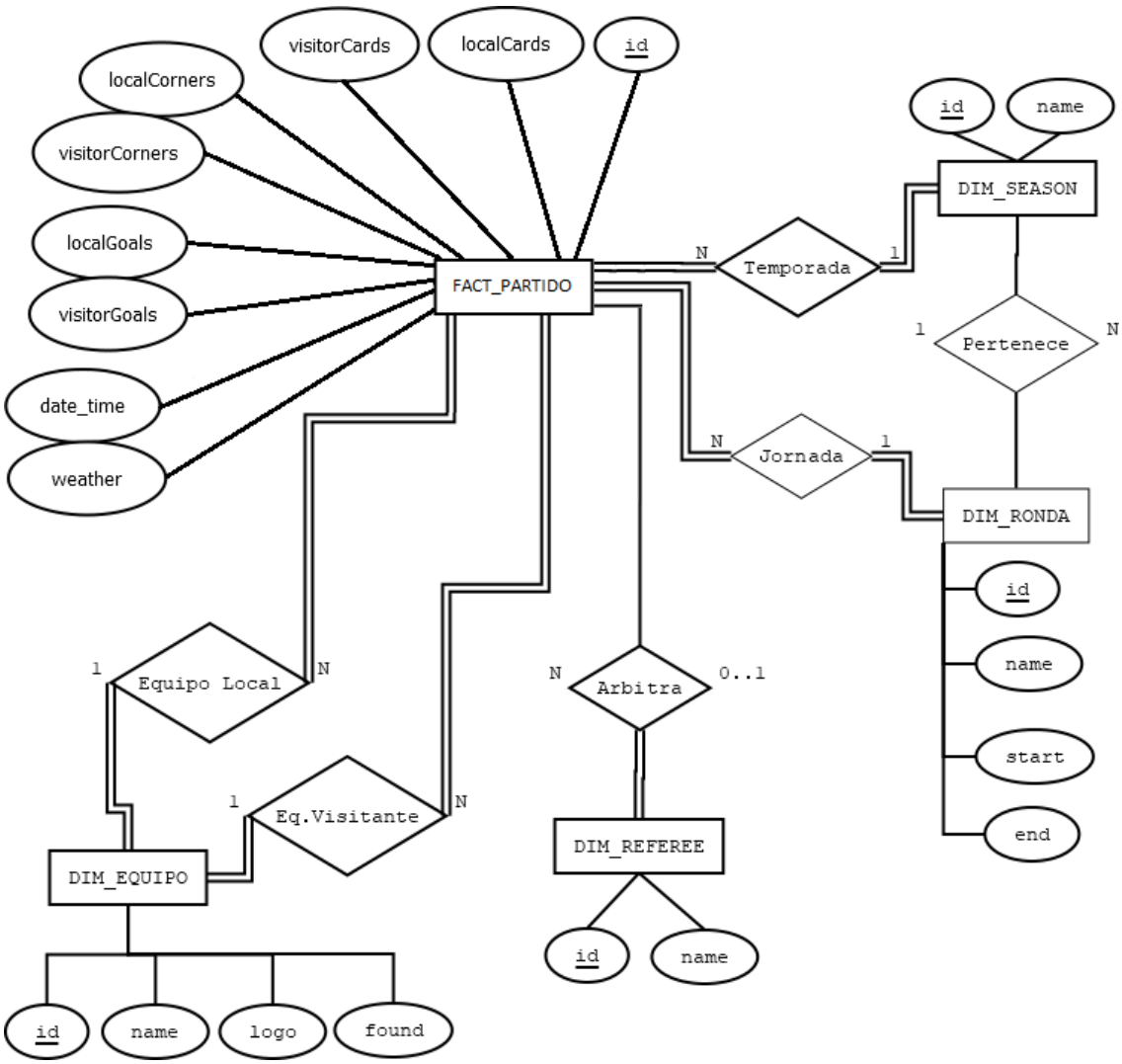


Figura 5.5: Modelo Entidad-Relacion BBDD.

Para una descripción ampliada de los atributos y entidades, a continuación mostramos el diccionario de datos aportando significado a nuestro modelo.

Diccionario de datos.

Tabla	Atributo	Descripción
FACT_PARTIDO		Tabla que contiene todos los partidos de las últimas 3 temporadas.
	id	id, clave del partido, autogenerado.
	localCards	Valor de las tarjetas correspondientes al equipo local.
	visitorCards	Valor de las tarjetas correspondientes al equipo visitante.
	localCorners	Número de córners del equipo local.
	visitorCorners	Número de córners del equipo visitante.
	date_time	Fecha y hora del encuentro.
	weather	Imagen del tiempo meteorológico en formato URL.
DIM_EQUIPO		Tabla que contiene todos los datos relativos a los equipos.
	id	id, clave del equipo.
	name	Nombre del equipo.
	logo	Imagen del escudo del equipo en formato URL.
	found	Año en que se fundó el equipo.
DIM_RONDA		Tabla que contiene todos los datos relativos a las jornadas.
	id	id, clave de la jornada.
	name	Nombre de la jornada.
	start	Fecha de inicio de la jornada.
	end	Fecha de finalización de la jornada.
DIM_SEASON		Tabla que contiene todas las temporadas.
	id	id, clave del equipo.
	name	Nombre de la temporada.
DIM_REFEREE		Tabla que contiene todos los datos relativos a los árbitros.
	id	id, clave del equipo.
	name	Nombre completo del árbitro.

Figura 5.6: Diccionario de datos

5.2.2 Diseño

Finalmente partiendo del modelo ER (Figura 5.5, diseñamos un modelo relacional.

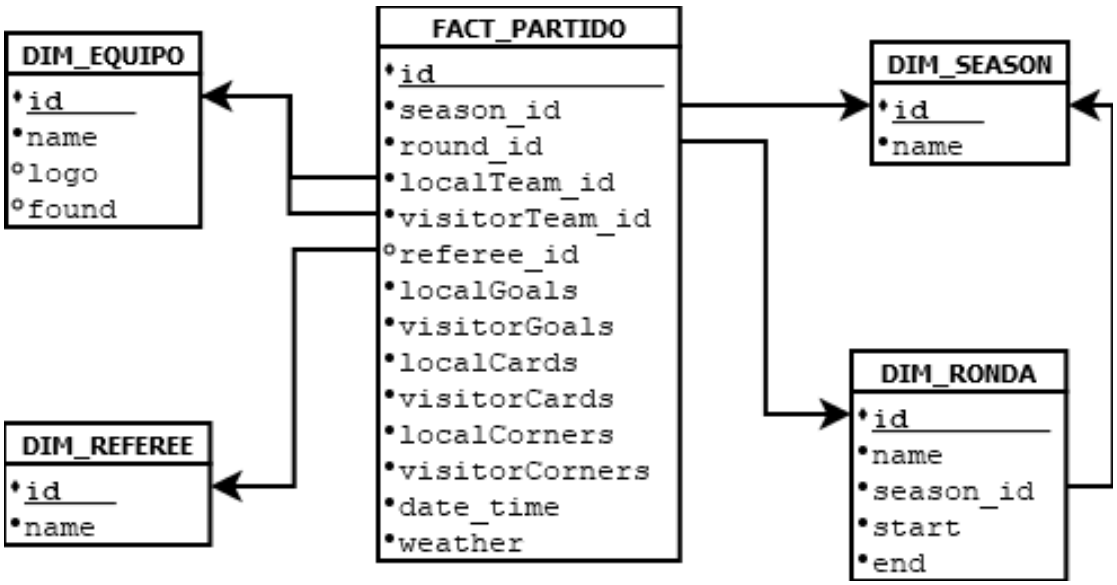


Figura 5.7: Modelo Relacional.

En la parte de la ETL, el diseño que planeamos se ve ilustrado en la siguiente imagen:

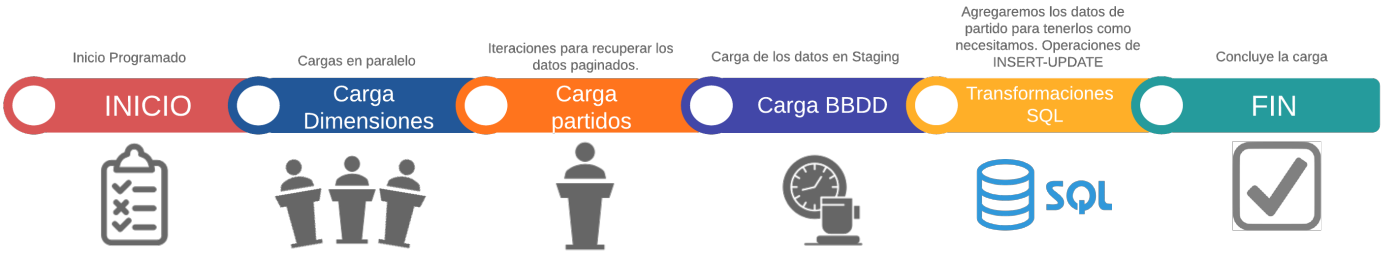


Figura 5.8: Diseño ETL.

De cara al staging se han definido 6 tablas para poder preparar los datos antes de insertarlos. Las hemos ubicado en un esquema propio y el usuario de ETL es quien tiene permisos totales sobre estas tablas.

Crearemos una para cada dimensión (4 en total) y dos para los partidos. Las de los partidos son: una primera donde volcamos los datos desde Pentaho directamente y otra que utilizamos como paso previo agregando la información del partido. El proveedor lo dispone a nivel de evento, es decir, cada tarjeta o córner; generaría una línea. Buscamos agregar de forma que tendríamos los partidos identificados por clave teniendo una única línea por partido.

Los objetos como el árbitro, tarjetas, goles y córners se encuentran contenidos dentro del JSON en otros objetos JSON, por lo que es necesario realizar transformaciones secundarias para poder subirlo al nivel del partido. Esto generará lineas adicionales respecto a lo que realmente queremos, pero es la única forma de pasar la información del modelo objetual que tiene el proveedor a un modelo relacional.

Para las temporadas, jornadas y equipos no necesitamos hacer transformaciones, nos llevamos los datos en bruto del proveedor. Estas cargas se lanzan antes de la carga de árbitros y partidos para que no nos falle la integridad referencial cuando carguemos los partidos.

En Pentaho las cargas se ubican dentro de Jobs. Dichos Jobs están compuestos de una o varias transformaciones. Pentaho considera una transformación incluso llevarse la información directamente, por lo que para evitar confundirnos cuando hablemos de transformaciones para Pentaho y transformaciones en el sentido de ETL, las de Pentaho las llamaremos mappings de ahora en adelante.

Los lanzamientos de carga los planificaremos en la madrugada, cuando menos gente pudiese estar utilizando la aplicación. No obstante no habrá corte en el servicio dado que al utilizar el Staging, el gestor de base de datos nos garantiza que hasta que se finalice la transacción no hay cambios visibles en la tabla.

5.2.3 Implementación:

Generamos las consultas DDL pertinentes para llevar el modelo relacional al Datamart. Damos de alta las restricciones foráneas necesarias para respetar la integridad del modelo.

Ahora que tenemos la base de datos lista para recibir información, empezamos a implementar en Pentaho todos los jobs y mappings. Por simplicidad, las dimensiones se cargan de forma completa, esto implica que en las dimensiones no guardamos ningún tipo de histórico. Tras cada carga ETL tendremos datos nuevos.

Una vez están los datos cargados en Staging, e identificados por clave, realizamos una operación de INSERT-UPDATE (MERGE en Oracle) contra la tabla de hechos por si algún valor cambiase. Los datos son susceptibles de cambiar porque las jornadas se publican semanas antes de acontecer, y los árbitros pueden cambiar, la meteorología no es fija, e incluso la fecha del partido puede verse alterada.

Hemos parametrizado en el Job principal la clave que nos identifica ante el proveedor (API Key) para conectarnos al él con nuestra identidad verificada, siendo de esta forma fácil de cambiar si fuera necesario. En cada mapping se ha de construir el endpoint contra el que hacemos la petición, después la realizamos y toda la información nos viene dada en una única respuesta en formato JSON que debemos hacer corresponder con los campos de nuestra tabla.

El resultado final de árbitros es una dimensión cargada con los datos de los colegiados que arbitrasen algún partido en las últimas 3 temporadas.

El resultado final de los partidos, al terminar el mapping en Pentaho, es una tabla de Staging donde la información esta a nivel de evento y no de partido. Es decir, esta a nivel de que cualquier tarjeta o córner genera una fila entera con toda la información del partido. Con los goles no ocurre esto porque el proveedor proporciona un campo que es justamente el resultado final. El último paso que realiza la ETL es agregar toda esa información a nivel de partido utilizando consultas SQL directamente sobre las tablas de Staging.

Nos hemos encontrado con el particular problema de la paginación de la respuesta por parte del proveedor. Esto se debe a que, al tener una volumetría tan elevada de datos (todos los partidos de las últimas 3 temporadas de La Liga, con eventos asociados), nuestro proveedor lo entrega paginado.

El tratamiento de paginado en Pentaho no es un caso contemplado por la aplicación, pero lo hemos solventado recuperando el campo “next-url” del JSON que nos envía el proveedor,

construyendo la siguiente petición adhiriéndole la API-KEY y los parámetros que utilizásemos, y volviendo a llamar hasta que el campo “next-url” sea nulo.

Para ilustrar un poco este proceso, mostramos el Job principal de Pentaho y explicamos sus áreas de manera que lo hagamos mas fácil de imaginar.

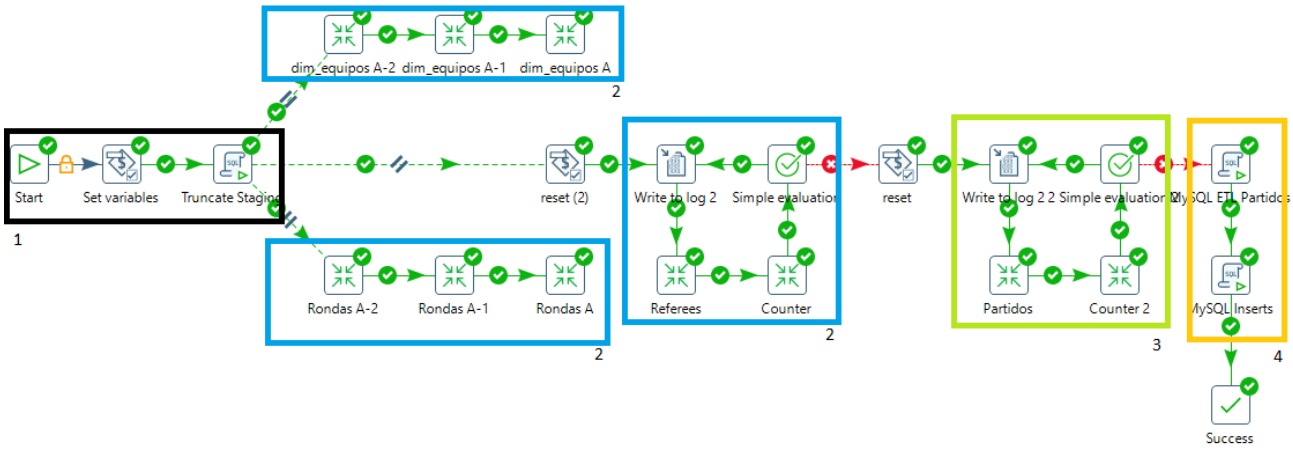


Figura 5.9: Job Principal en Pentaho.

Dividimos el Job en 4 regiones.

1. Zona Negra - Se realiza el inicio, truncado del staging y asignación de variables como fechas y la API KEY
2. Zona Azul - Carga en paralelo de todas las dimensiones.
3. Zona Verde - Carga de partidos, debe iterar por paginado.
4. Zona Naranja - Carga de las tablas FACT desde el Staging, se hace con llamadas SQL sobre MySQL.

Hemos definido tareas recurrentes en el servidor para que se lance una vez a la semana una carga completa, dicha tarea se realiza los miércoles porque es donde menos incidencia de partidos de La Liga hay. Definimos también una carga para el resto de días, durante la madrugada a las 6:00am porque es cuando se estima que tendríamos menos impacto dado el bajo uso de la app. Esta tarea se realiza mediante el Programador de Tareas de Windows, lanzando un script que llama a la tarea en Pentaho.

Implementaremos a continuación las vistas de datos.

Vista de estadísticas

Se definen vistas de las estadísticas donde calculamos la media de cada equipo como local y como visitante. Siempre nos referimos a las estadísticas relativas a nuestras 3 apuestas objetivo: Goles, Tarjetas y Córners.

Vista de árbitros

Se define una vista de los árbitros sobre tarjetas, creemos que saber la tendencia que sigue un colegiado a la hora de decidir si amonestar o no, nos afinará nuestra predicción.

Vista de totales

Se define una vistas de totales donde calculamos los datos generales de los locales y visitantes, sin bajar a nivel de partido. Necesitamos esto para calcular las Fuerzas que explicamos en el modelo matemático.

Vista de predicciones

Se define la vista de predicciones, esta vista es el elemento clave que necesita la aplicación para funcionar, ahí agruparemos todos los datos que va a necesitar mostrar la aplicación. Hacemos los cálculos acorde al diseño previo. Aquí es donde unificamos todos los conceptos del modelo matemático y realizamos la ecuación 5.4 para recuperar las probabilidades sumadas.

Explicamos a continuación nuestro cálculo estadístico para la predicción. Este mismo varía dependiendo de si calculamos goles, tarjetas o córners.

- Para los goles, se calcula la probabilidad sumada de gol siguiendo una distribución de distribución de Poisson para local y visitante.
- Para los córners se hace lo mismo que para los goles.
- Para las tarjetas tenemos en cuenta varios factores. Recuperamos el valor medio de tarjetas de un equipo como local y visitante pero a mayores debemos tener en cuenta que si el árbitro es alguien que amonesta bastante, nos va a introducir un error.

  Para tener en cuenta la parte del árbitro, hemos calculado la media de amonestaciones por partido del árbitro, y su desviación respecto a la media de todos los árbitros. También hemos aprovechado este cálculo para hacer un ranking de los colegiados en función de su media de amonestaciones por encuentro.

Realizamos nuestra predicción sobre tarjetas utilizando la media del equipo en su respectiva posición (local/visitante) y sumándole la mitad de la desviación media del árbitro.

Hemos tomado esta decisión de cálculo basándonos en la observación. En las pruebas, en concreto en la figura 5.11, mostramos diversos resultados alternando el peso del árbitro.

La aplicación necesitará, aparte de todo lo anterior, mostrar los datos de partidos previos. Para los partidos anteriores no haría falta crear una vista, directamente se puede acceder por fecha contra FACT_PARTIDO porque ya tenemos los datos materializados ahí. Sin embargo hemos creado una vista que accede a esa tabla sin ningún tipo de lógica adicional por respetar el modelo ANSI-SPARC.

5.2.4 Pruebas

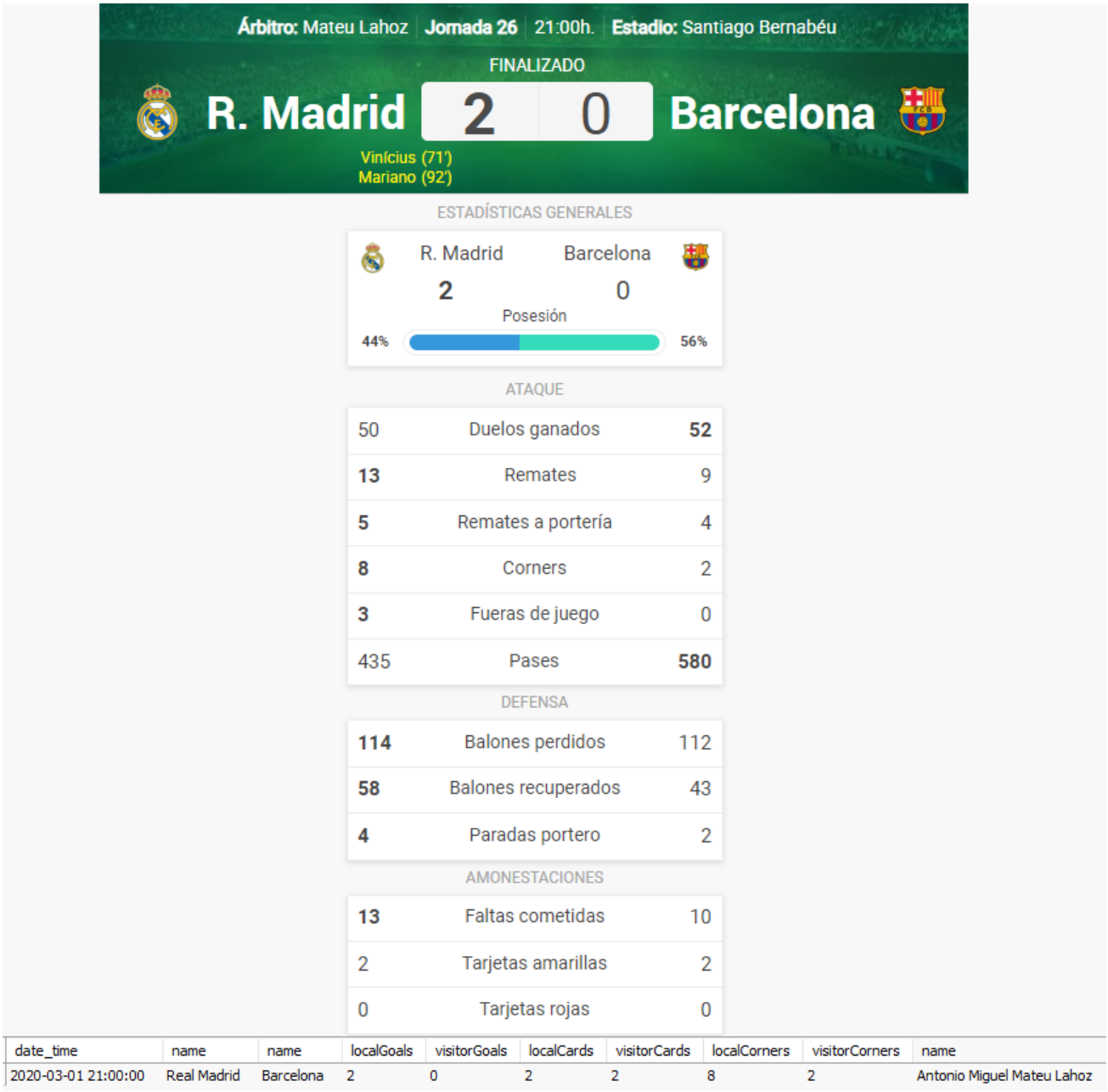
Para las pruebas hemos comenzado probando de manera individual las partes desarrolladas en esta iteración, y finalmente las hemos probado en conjunto.

En la parte de Pentaho se establecieron pruebas unitarias comprobando en que tras cada transformación los datos continuaban el flujo y sufrían los cambios que se pretendían conseguir. Finalmente se probó que se escribían los datos correctos en las tablas destino.

Por parte de la base de datos se probó la integridad del modelo intentando vulnerar las restricciones foráneas obteniendo los errores esperados. Probamos también descargando la información del servidor en formato JSON y comprobando si una información concreta estaba presente y correcto en nuestra base de datos.

Finalmente para las pruebas de integración, hemos lanzado una carga completa y comprobamos que los datos que se han llevado a la base de datos son fidedignos. Para poder probar esto nos hemos servido de la hemeroteca de `Marca.com`. Hemos seleccionado algunos encuentros de forma aleatoria, y hemos comprobado los datos siguiendo el siguiente método:

Seleccionamos un partido concreto y vemos si los resultados y el colegiado que figuran en el website se corresponden con los de nuestra base de datos. Como podemos observar, coinciden exactamente.



date_time	name	name	localGoals	visitorGoals	localCards	visitorCards	localCorners	visitorCorners	name
2020-03-01 21:00:00	Real Madrid	Barcelona	2	0	2	2	8	2	Antonio Miguel Mateu Lahoz

Figura 5.10: Pruebas ETL. Arriba Marca.com, abajo nuestra base de datos.

Para poder probar la calidad de nuestras predicciones hemos creado una vista específica para medir el error pronosticado contra el valor real que ocurrió en su día en el estadio.

La vista nos devuelve los errores medios de nuestros pronósticos con un acceso simple. Es muy cómoda para poder realizar este tipo de pruebas.

La calidad de la predicción la hemos afinado bastante. No obstante, siempre existe un error insalvable que es la propia entropía del fútbol. Existen muchas variables a tener en cuenta y existen diferentes factores que no podemos calcular o ponderar para corregirnos.

Tras aplicar la Distribución de Poisson 5.1, los errores finales calculados con la vista de calidad son:

- Goles : 1.287017
- Córners : 3.544686

Para las tarjetas hemos ido ajustando el peso de la desviación típica de los árbitros al calcular las tarjetas en las predicciones, se ha comprobado que, aplicando dicha desviación, el error medio en las tarjetas ha bajado un 67%.

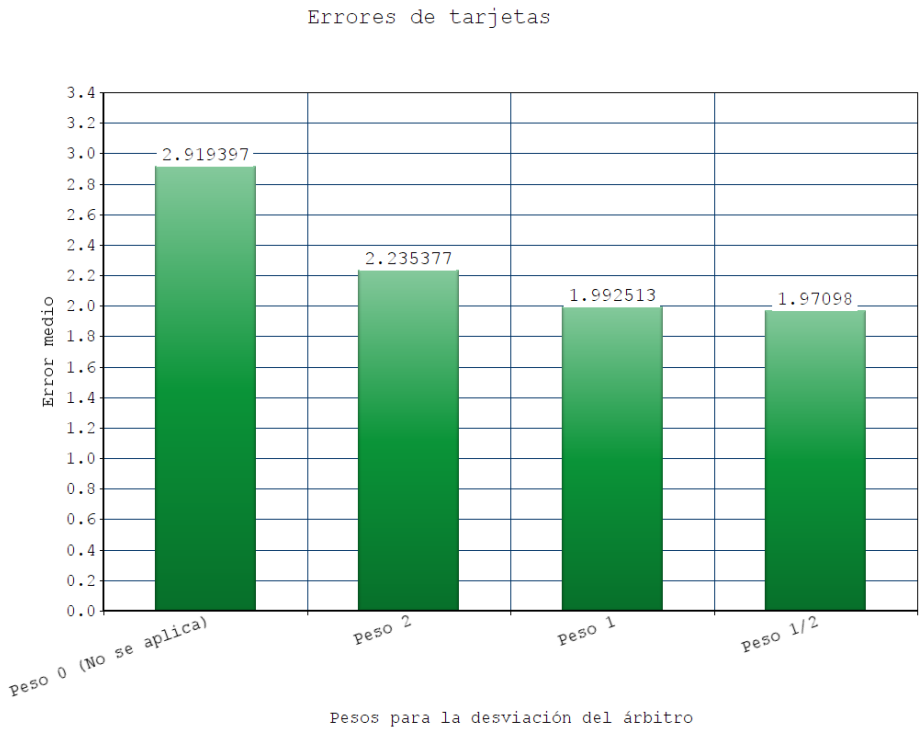


Figura 5.11: Pruebas calidad en tarjetas.

Se han realizado pruebas de rendimiento que adjuntaremos en los apéndices. A.1

Otra conclusión de estas pruebas es que vemos que la ETL no hace falta lanzarla con mucha regularidad, los datos históricos no suelen cambiar. Estamos programando una carga incremental al día y una total, con periodicidad semanal, que se haría los martes porque ya terminaría la jornada anterior. Entendemos por carga total una carga de todos los partidos de las últimas 3 temporadas y por incremental cargamos los últimos 30 días. Dado el poco tiempo que le lleva al proceso, no vemos problema por realizar las cargas que fueren necesarias. Hablamos de que en el caso mas pesado no llega a 4 minutos.

5.3 Iteración 2 - Webservice API REST

El objetivo concreto de esta iteración es el desarrollo necesario para crear un servicio REST que exporte la información de nuestra base de datos a internet.

5.3.1 Análisis.

Es necesario mostrar 4 grupos de información para alimentar las pantallas que planeamos en la aplicación móvil:

1. Los datos referentes a un partido en concreto. El partido es la entidad que nos relaciona todo: equipos, árbitro, meteorología y fecha del encuentro.
2. Los datos referentes a los equipos, recuperaremos las estadísticas de los equipos y sus escudos (logos).
3. Los datos referentes al árbitro, necesitaremos recuperar su nombre y su puesto en el ranking de amonestaciones.
4. Los datos de los partidos históricos. Estos valores los queremos para dar retroalimentación al usuario; es decir, daremos los hechos que se produjeron pero no entraremos a contrastarlos con nuestras predicciones pasadas.

Para cumplir con todo lo anterior, vemos que nuestros requisitos son:

- El servicio debe estar accesible desde el exterior de la red.
- Tenemos que proporcionar información de los próximos partidos desde una fecha proporcionada.
- Tenemos que proporcionar información de los partidos históricos para 2 equipos concretos proporcionados.
- El servidor REST tiene que transformar el valor relacional en diccionarios JSON.

5.3.2 Diseño.

Para representar la información utilizaremos DTOs (Data Transfer Object) y para el acceso a datos utilizaremos el patrón DAO (Data acceso object).

Ilustramos el patrón en la siguiente imagen.

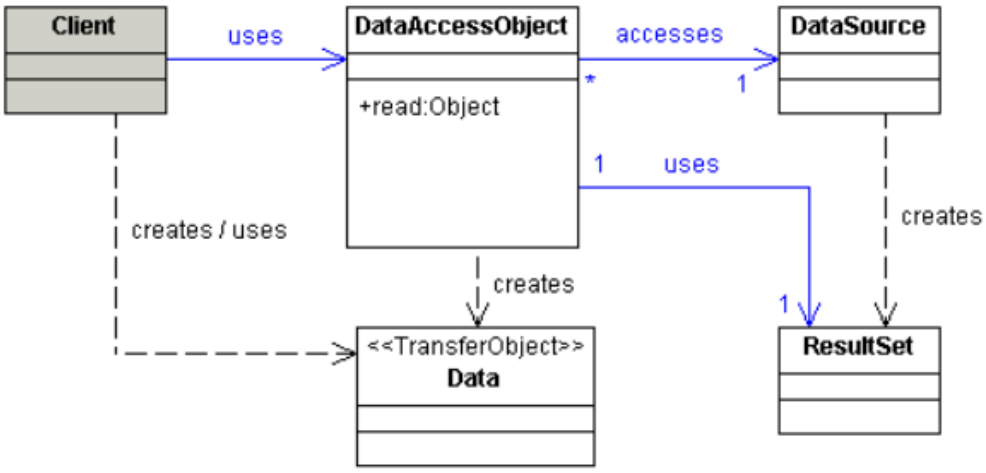


Figura 5.12: Patrón DAO. [2]

Un DAO es un adaptador entre la lógica de negocio y la base de datos; por tanto, su labor es la de ser un traductor entre nuestra aplicación y la base de datos. El objetivo de este adaptador es codificar la lógica de acceso a los datos.

Esto implica que por cada entidad tenemos que implementar un DAO específico. En nuestro caso las entidades serían: el partido, el equipo, el árbitro y el partido pasado porque que tiene una estructura diferente al actual, no recuperamos los mismos campos.

Los DTO que hablamos anteriormente son una clase intermedia que se encarga de representar como un objeto la estructura de la tabla de una base de datos.

La estructura de paquetes que planeamos creado es la siguiente:

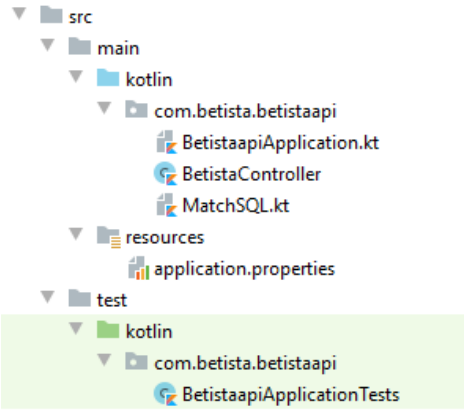


Figura 5.13: Estructura de paquetes.

Utilizando Spring nos evitamos tener que crear todos los servlets y con anotaciones ya le indicamos el comportamiento a seguir. Hemos creado los ficheros que se ven en la figura 5.13 y los explicamos a continuación:

- El fichero BetistaapiApplication.kt es el constructor e iniciador del servicio.
- En BetistaController definimos los parámetros de conexión a la base de datos y los mapas para que Spring sepa a donde redireccionar las peticiones y que consultas lanzar, aquí están implementados los DAO.
- En MatchSQL.kt implementamos los DTO.
- El fichero que se encuentra abajo separado y marcado en verde son las pruebas de JUnit.

Existe una equivalencia entre los DTO y la estructura relacional.

Dado que tenemos una base de datos hecha exclusivamente para esta aplicación, con solo dos endpoints podemos recuperar toda la información que requiere la app. Definimos entonces los endpoints:

/upcomingMatches

Esta llamada ya recupera la información necesaria para construir los objetos de partido, de equipos y de árbitro. Estamos aprovechando la llamada inicial para enviarle todos los datos de esos 20 partidos siguientes a mostrar, hablamos de las estadísticas predictivas, el colegiado, los datos de los equipos...

Lo único que no estamos enviando son los resultados anteriores porque sería muy pesado enviar todos los históricos. Así que diseñamos otro endpoint únicamente para los partidos pasados.

Lo ilustramos con un diagrama de secuencia, el otro endpoint obviamos ilustrarlo porque es prácticamente lo mismo cambiando el DTO y la consulta.

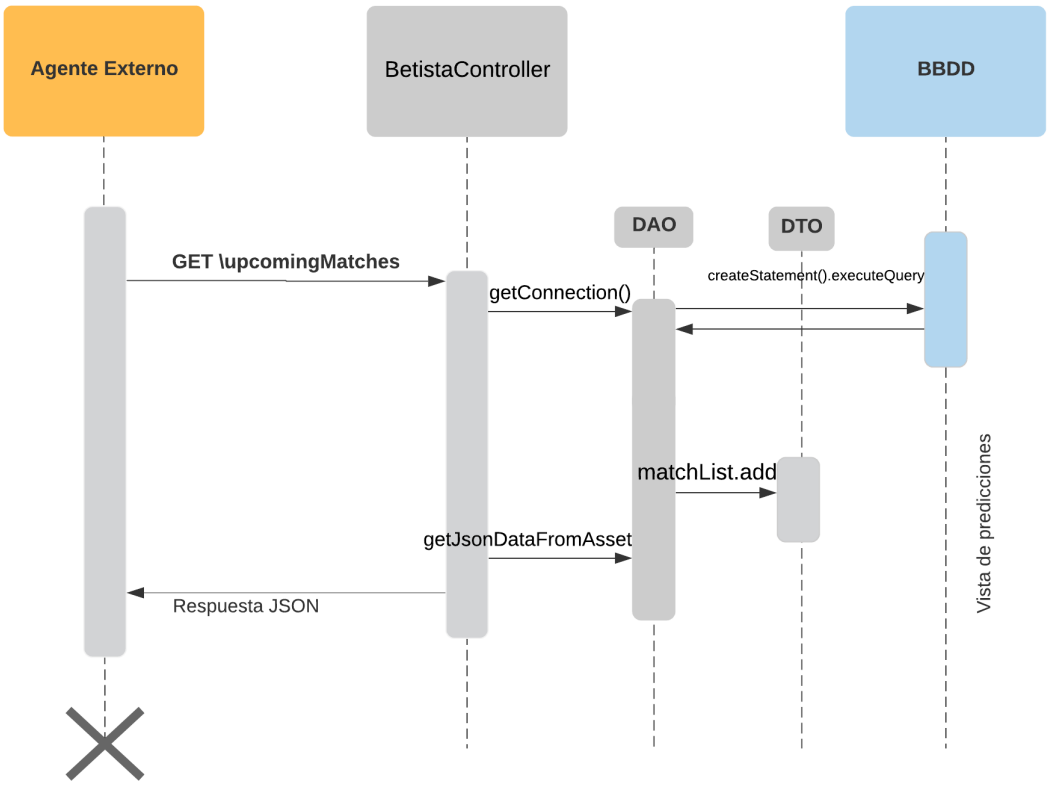


Figura 5.14: Secuencia de endpoint de partidos.

La API REST lanzará la consulta definida en el DAO contra la base de datos, recuperara la información y construirá los DTO. Utilizando estos DTO preparará una respuesta en JSON que devolvera a la dirección que hizo originalmente la solicitud.

/past

La idea es que al seleccionar un partido de los que mostremos, el usuario podría querer ver los últimos enfrentamientos entre estos equipos.

Por motivos de presentación, se ha añadido un parámetro de fecha para este endpoint. Es debido a que en la situación en la que se desenvuelve este TFG se encuentra pausado el fútbol profesional. La fecha se añadirá en la aplicación mediante un selector al inicio de la APP.

Este endpoint recibe como parámetro los ids de equipo, que recuperamos en la primera llamada, para consultar los resultados de enfrentamientos previos entre estos dos equipos. A mayores y dado que estamos permitiendo simular fechas anteriores, es necesario pasarle dicha fecha para poder filtrar y no mostrar resultados de partidos que serían futuros para esa fecha.

La configuración de red nos exige que tengamos el servidor activo las 24 horas del día. Configuraremos una dirección DNS por si nuestro proveedor de red cambiase nuestra IP por algún reinicio o pérdida de servicio puntuales.

En nuestro firewall, debemos redirigir las peticiones hacia nuestros endpoints hacia el servidor REST y dejar inaccesible cualquier otro acceso. No nos ha surgido la necesidad de crear un acceso de mantenimiento administrativo en remoto.

5.3.3 Implementación

Siguiendo los patrones de diseño elegidos, implementamos nuestro servicio REST con la estructura de paquetes de la figura 5.13. Hemos codificado un método extra, a los necesarios para implementar el patrón, que nos sirve para devolver la respuesta en formato JSON, que es el formato de entrada de datos que manejaremos en la aplicación móvil.

5.3.4 Pruebas

Hemos realizado pruebas unitarias y de integración utilizando JUnit 5 para Kotlin desde el propio Android Studio.

En las pruebas unitarias hemos probado que el servicio recupera correctamente valores de la base de datos y que construye bien los objetos.

También hemos probado que recibe correctamente las peticiones de red externas.

Probamos también que los datos que tienen los objetos construidos se corresponden con los de la base de datos.

Para probar completamente el sistema, donde podemos ver también que la construcción de los diccionarios JSON de respuesta son correctos, hemos realizado peticiones desde el propio navegador. Lo hemos hecho llamando directamente al servicio, pasándole los endpoints con los parámetros adecuados:

Partidos como si la petición se hiciera en el momento 2020-02-10 01:13:20.0000

`http://tryker.no-ip.org:8080/upcomingMatches?currentDate=1581293600000`

Partidos anteriores, a la fecha de la petición, entre el Celta de Vigo y el Barcelona.

`http://tryker.no-ip.org:8080/past?localId=36&visitorId=83&date=1581293600000`

Contrastando los resultados de los JSON contra la base de datos vemos que la información es coherente.

5.4 Iteración 3 - Aplicación Android

El objetivo concreto de esta iteración es crear una aplicación móvil que permita servir de frontal para mostrar los datos de nuestros pronósticos.

5.4.1 Análisis.

Crearemos una aplicación que se conecte a la API REST que diseñamos en la anterior iteración.

No sabemos el dispositivo móvil concreto que tendrán nuestros usuarios, así que usaremos una versión moderna de Android para llegar a un número elevado de personas.

Necesitamos desarrollar una serie de pantallas que muestren los valores pronosticados para los siguientes partidos de La Liga.

Las principales áreas de la aplicación son la vista inicial donde el usuario llega nada mas abrir la APP, la vista al seleccionar una predicción, la vista de estadísticas y la vista de partidos históricos.

Como requisito no funcional anotamos también que el usuario podrá seleccionar una fecha concreta para ver las predicciones de ese momento. En una versión futura este requisito desaparecería, no es interesante que pueda ver predicciones pasadas.

5.4.2 Diseño

Para llevar a cabo la aplicación seguiremos un patrón MVP (Model view Presenter). Es una variación del MVC (Model view Controller) convencional.

Hablaremos un poco de los dos y de por qué hemos elegido MVP para nuestra aplicación.

El MVC es un patrón de arquitectura de software que separa en tres bloques las partes representativas de la aplicación:

- Los datos y la lógica.
- La interfaz de usuario.
- El módulo encargado de gestionar los eventos y las comunicaciones.

MVC se compone entonces de tres partes distintas que son el modelo, la vista, y el controlador; es decir, de un lado tenemos componentes para la representación de la información y en otro lado la gestión de la interacción con el usuario. La base de este patrón son las ideas de reutilizar el código y de separar los conceptos clave.

MVC busca facilitar el desarrollo y el posterior mantenimiento de las aplicaciones.

Ilustraríamos MVC de la siguiente manera:

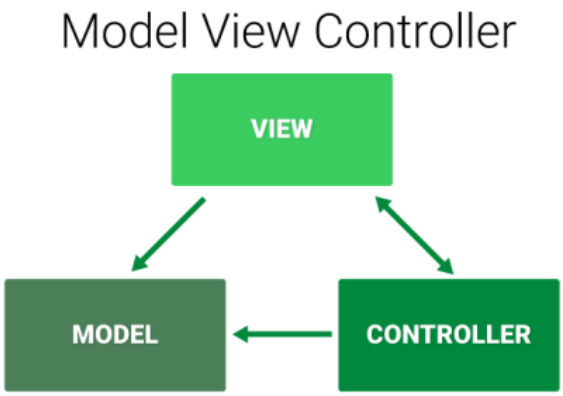


Figura 5.15: Modelo Vista Controlador.

El MVP es una derivación del patrón arquitectónico modelo-vista-controlador (MVC), es utilizado mayoritariamente para construir interfaces de usuario.

El comportamiento de las tres partes es el siguiente:

- El modelo es simplemente una interfaz que define los tipos de dato a mostrar.
- La vista es una interfaz pasiva que simplemente muestra los datos y enlaza las acciones del usuario (eventos) al acto que hay detrás de cada dato.
- El presentador toma la funcionalidad de mediador, actúa tanto sobre el modelo como sobre la vista, recupera datos de los repositorios (que ofrece la interfaz del modelo) y los transforma para alimentar a la vista asumiendo el peso de toda la lógica de presentación.

Ilustraríamos MVP de la siguiente forma:

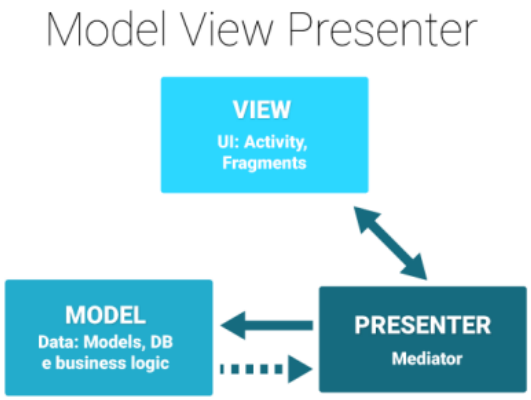


Figura 5.16: Modelo Vista Presentador.

Las principales diferencias son:

1. En MVC, cuando el modelo sufre un cambio de estado tiene la obligación de notificar de ello a la vista. La información del cambio puede pasarse en la misma notificación o después, la vista en todo momento podría consultar al modelo directamente para obtener un refresco de datos. En MVP, en cambio, la vista no sabe nada sobre el modelo y para actualizar sus datos debe consultar al presentador, cuya labor es mediar entre ellos.

2. En MVC, la vista generalmente tiene más lógica porque es quien se encarga de manejar las notificaciones del modelo y de trabajar los datos. En MVP, esta lógica se implementa en el presentador, haciendo a la vista muy simple. La única labor de la vista es representar la información que el presentador le brinda.

3. En MVC, el modelo tiene que implementar lógica extra para comunicarse con la vista. En el MVP, ese trabajo recae sobre el presentador.

Continuamos nuestro diseño con las pantallas. Nos basaremos en los mockups que planificamos anteriormente. Definimos 4 vistas para la aplicación:

- **La vista de partidos**, una parrilla con los próximos 20 partidos (el número es por limitar datos, pero cuadra con los partidos que hay en las próximas 2 jornadas).

- **La vista de una predicción**, a la que se accede desde la vista de parrilla seleccionando con una pulsación.

- **La vista de partidos pasados**, a la que se accede desde la vista de una predicción, seleccionando que enfrentamiento anterior se desea ver

- Y finalmente **la vista de estadísticas**, donde mostramos las estadísticas pronosticadas para ese partido, se accede desde la vista de predicción.

Queda ilustrado en el siguiente diagrama de flujo:

Figura 5.17: Diagrama de Flujo

5.4.3 Implementación.

Diseñaremos en Android Studio una aplicación siguiendo el patrón Model-View-Presenter.

A mayores de las librerías de Google, incluidas con Android Studio, vamos a utilizar las siguientes:

- GSON, utilizada para parsear datos de un json que viene de la red.
- Retrofit, con su conversor de Gson, para realizar las peticiones de red.
- Material, buscando dar una apariencia mas moderna a la app, es una librería de diseño.
- Picasso, que nos permite a partir de una URL cargar su imagen asociada, para los logos de los equipos.
- JUnit 5, para las pruebas
- RecyclerView que es para poder tener listas dinámicas y actualizables mientras se muestran en una vista.

Para generar los diseños nos vamos a servir de la utilidad gráfica que nos trae Android Studio.

Dicha utilidad nos permite pintar los XML que formarían las vistas de manera interactiva para evitar tratar tan directamente con el código, aunque hemos tenido que bajar a ese nivel en repetidas ocasiones para matizar cuestiones de diseño menores.

Es particularmente interesante para hacer pequeños cambios como desplazamientos de objetos en al diseñar la pantalla sin tener que perder mucho tiempo calculando manualmente por píxeles o tamaños relativos.

En los apéndices pueden encontrarse los diseños XML realizados.

En la siguiente página mostraremos el esquema de paquetes del proyecto y explicaremos cada clase por encima para ilustrarlas.

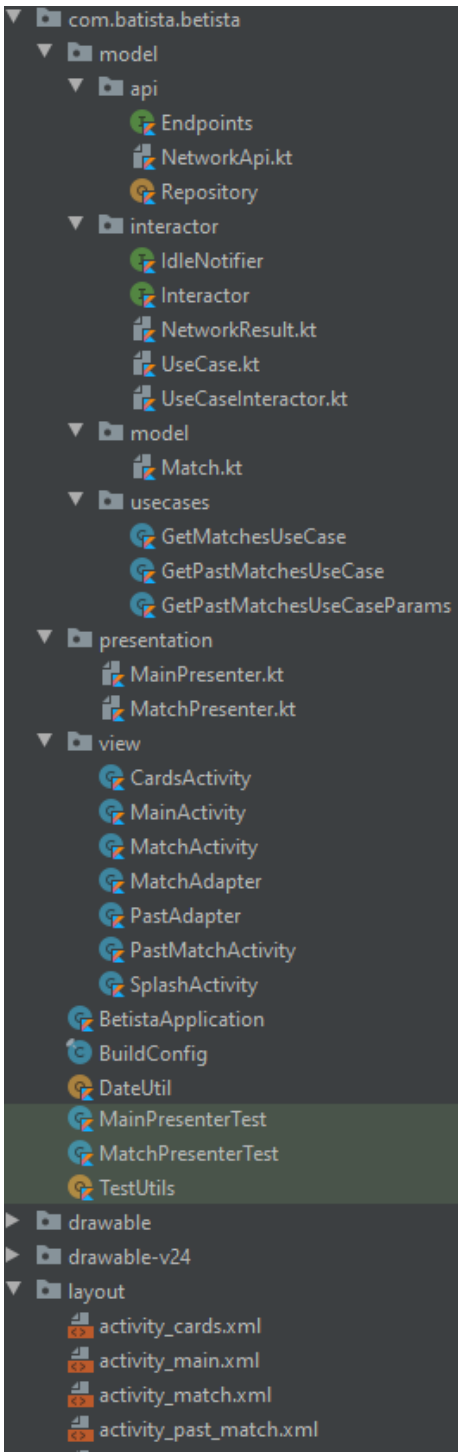


Figura 5.18: Esquema del proyecto.

Siguiendo el esquema del proyecto android:

En Api definimos los endpoints de la app. /upcomingMatches (que recordamos necesita recibir una fecha) y /past (que recibe los ids de los equipos además de la fecha).

Los objetos del interactor (user cases) los definimos en el modelo. Representan las acciones posibles del usuario.

En Match.kt se implementan los objetos de almacenamiento de datos.

En presentation definimos la lógica y los métodos a llamar para hacer los cambios visibles en la app.

En view tenemos las diferentes vistas. La vista no conoce el modelo, y su comunicación es siempre a través del presenter.

El presenter llama al interactor (model), este a su vez llama a la API y recibe los datos, luego notifica al presenter que finalmente notifica a la vista que se modifica. Es un diseño muy lineal.

Los ficheros marcados en verde son las pruebas de JUnit.

En layout tenemos el diseño XML de cada una de las pantallas y sus componentes internos.

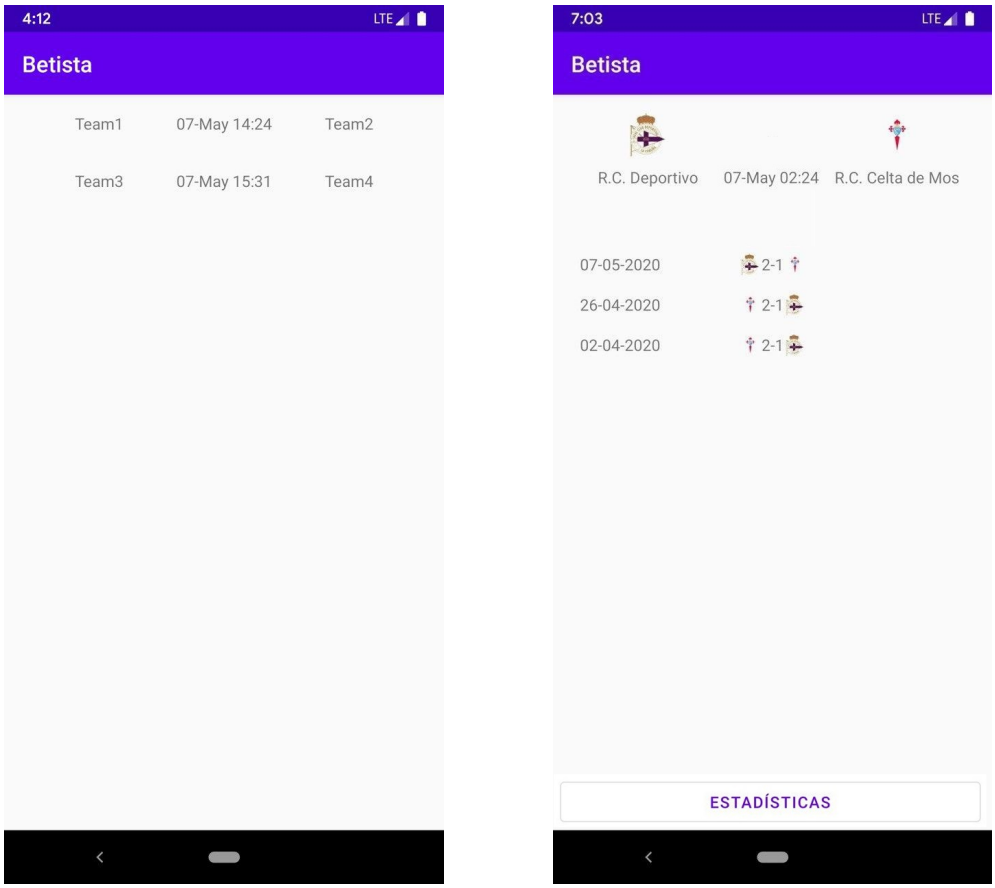
5.4.4 Pruebas:

Hemos comenzado realizando pruebas unitarias para probar que todas las clases realizan su cometido. Las realizamos con JUnit 5 centrándonos en probar las funcionalidades individuales, que al crear objetos desde los datos JSON se crean correctamente, que cuando se pulsa en los botones de una ventana se llama al método que abre la nueva vista, etc.

Hemos conseguimos cobertura bastante total de los métodos que utiliza la aplicación.

Las siguientes pruebas son de integración y ya nos centraremos en ir probando las diferentes interacciones entre los sistemas de todo el proyecto.

En un primer paso probaremos la integración de la aplicación en concreto, la realizamos añadiendo un JSON con datos de prueba para no involucrar de momento a las llamadas de red (API REST) y veremos las pantallas que genera la app a partir de esos datos.



(a) Vista de partidos.

(b) Vista de predicción

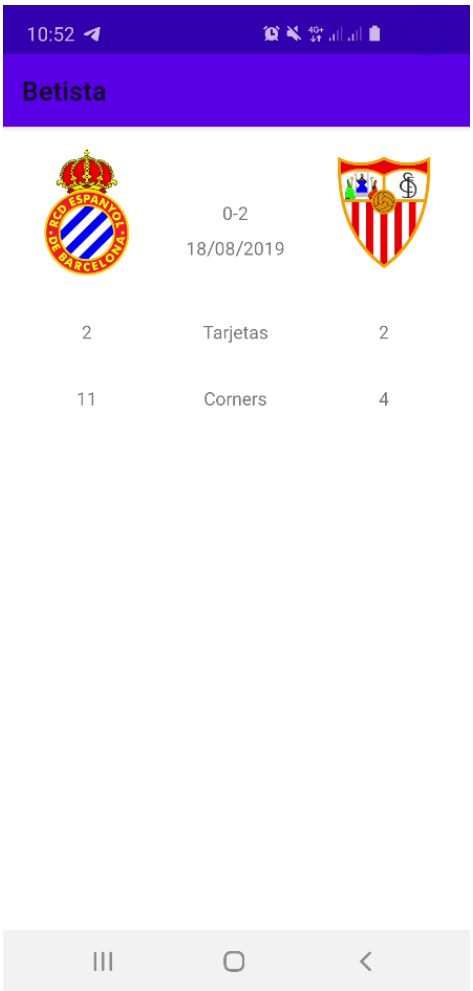
Figura 5.19: Pruebas de integración, sin red.

Probaremos ahora todo el sistema, vamos a incluir llamadas de red para recuperar valores de la API REST.

Seleccionamos partidos aleatorios y con el selector de fechas nos posicionamos un día antes del partido objetivo, de esta forma podemos ver las predicciones. Para ver los resultados que se produjeron y probar al mismo tiempo la vista de partidos pasados, elegimos la fecha donde se juegue el mismo partido pero en la segunda vuelta de La Liga (Local y visitante intercambian posiciones), de esta forma el partido para el cual vimos las predicciones figurara como un partido pasado.

A continuación mostramos solo un encuentro pero hemos probado con varios, esta no es la mejor de nuestras predicciones ni tampoco es la peor.

(a) Pronóstico.



(b) Realidad.

Figura 5.20: Prueba Real.

Analizando los resultados:

- Vemos que tanto las tarjetas amarillas del Espanyol como las del Sevilla han sido las que esperábamos. Por esa parte ha ido genial.
- Los córners por lo contrario no han ido tan bien los del Espanyol. Pero si que hemos acertado los del Sevilla.
- La parte de goles, que es la mas inestable acorde a nuestras pruebas, nos permite saber que el equipo favorito para ganar el encuentro era el Sevilla.
- Las apuestas a linea de Gol y número de tarjetas las hubiéramos acertado pero no la de córners.

Capítulo 6

Conclusións

En este último capítulo de la memoria hablaremos de qué hemos aprendido con este proyecto, qué objetivos se han cumplido y cuáles posibles líneas futuras tendría la continuación de nuestra idea.

Este proyecto nació de la falta de existencia de una herramienta gratuita para pronosticar apuestas deportivas en base a la matemática y no a la subjetividad inherente de las opiniones personales, que son fruto de la experiencia (tampoco pretendemos desmerecerlas).

Nuestro principal objetivo fue pronosticar apuestas deportivas siguiendo un modelo estadístico y lo hemos cumplido. Hemos aplicado la distribución de Poisson siendo fieles a su definición y el resultado es el esperado, tenemos bastantes predicciones donde acertamos y también algunas donde no, pero estas predicciones falladas nos ayudan a que en el futuro podamos acercarnos a tener todavía más tasa de acierto dado que nuestros pronósticos se basan en los resultados previos.

Con respecto a las herramientas utilizadas, hemos profundizado mucho en nuestro conocimiento sobre ETLs, especialmente sobre Pentaho, dado que en la carrera no pudimos explotar mucho la herramienta por falta de tiempo.

Hemos repasado conceptos de interfaces de usuario y hemos ampliado nuestro conocimiento en el desarrollo de aplicaciones a un nuevo lenguaje (Kotlin) que esta cobrando mucha importancia entre las empresas del sector a día de hoy, cada vez hay más aplicaciones utilizando el lenguaje dado que Google lo pone al mismo nivel que a Java para desarrollar Android.

Haciendo un repaso general al sector de las apuestas hemos podido concluir una idea que nos rondaba desde un principio, es muy difícil hacerse rico en este sector sin ser la banca. Ningún modelo matemático nos va a hacer de oro porque si no todo el mundo lo usaría y no habría negocio para las casas de apuestas. Nuestro modelo acierta bastante pero donde acertamos ya nos damos cuenta de que las cuotas no son tampoco muy lucrativas, dado que las casas de apuestas también pronostican y ponen cuotas bajas a resultados ya esperados.

Como líneas futuras del proyecto:

- A corto plazo podría explotarse más en profundidad la distribución de Poisson para conseguir predicciones a resultados fijos.
- Si pensamos en un plazo medio, ahí ubicaríamos añadir nuevas competiciones, no ceñirnos solo a La Liga si no a las diferentes ligas de todos los países. También se podría bajar a nivel de pronosticar teniendo en cuenta jugadores concretos y no solo equipos, a partir de una alineación determinada conseguir apuestas buscando apuntar hacia la apuesta Live (en tiempo real).
- A largo plazo, otra línea de investigación sería pronosticar otros deportes, aunque habría que replantearse si Poisson es el modelo que mejor se ajusta.

Mis sinceros agradecimientos por su tiempo a todos los que se han interesado y leído hasta aquí.

Apéndices

Apéndice A

Material adicional

A.1 Pruebas de rendimiento

Mostramos a continuación los tiempos de ejecución de la ETL:

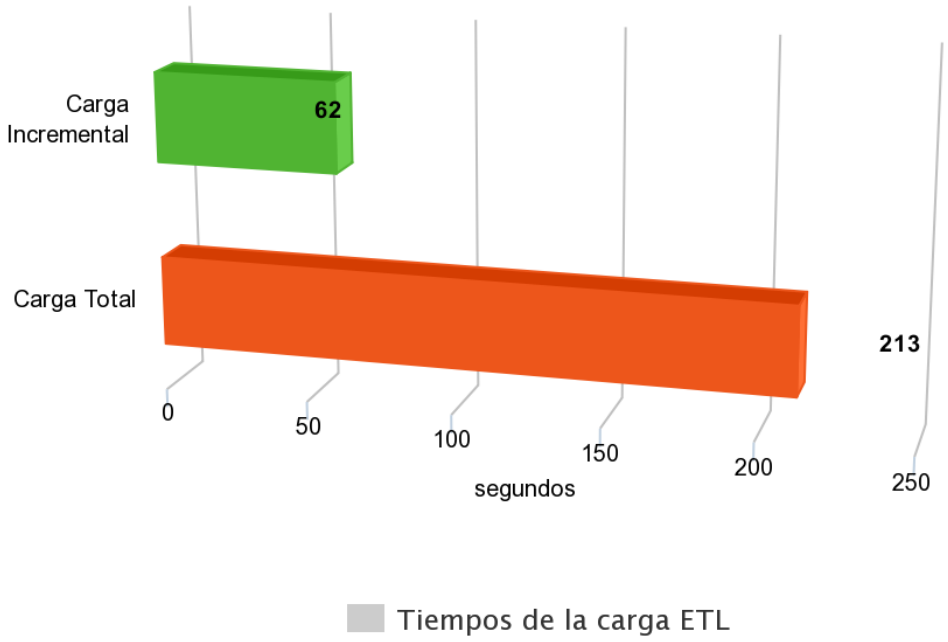


Figura A.1: Tiempos de carga ETL.

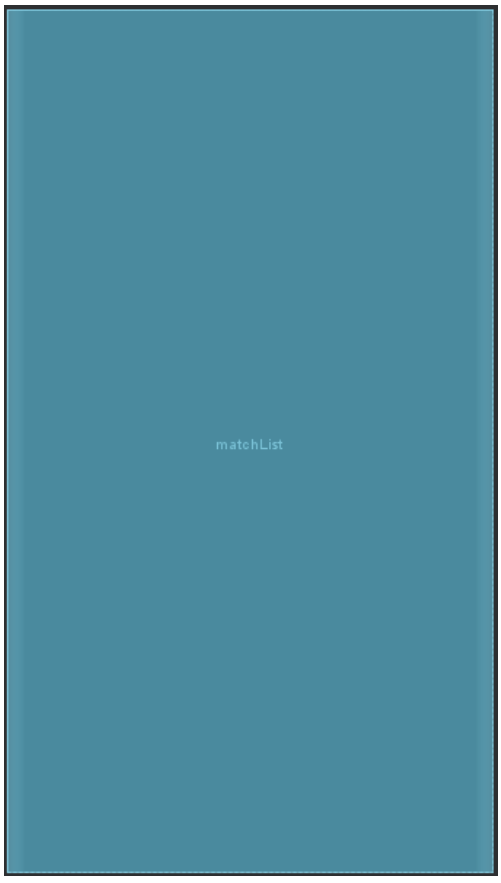
Como podemos observar no es un proceso largo. Podría realizarse una carga total a diario sin problema. Diseñamos el sistema con incrementalidad de cara a que, si añadimos otras ligas, el peso de la carga pudiera ser elevado. Nuestro proveedor tiene limitadas las consultas que podemos lanzar en cada plan de pagos. De esta forma estamos siendo prevenidos y podría significar un ahorro en caso del escalado a varias competiciones.

A.2 Diseño de Pantallas

Para cada pantalla mostraremos su diseño gráfico XML desde Android Studio y el resultado final de dicha pantalla.

Vista de Partidos

Esta es la vista principal que el usuario se encuentra al iniciar la aplicación. En ella figuran los próximos 20 partidos que se jugarán en la temporada actual de La Liga. El diseño XML es muy simple dado que solo mostramos el objeto de la lista. Si el usuario selecciona un partido el Presenter es notificado e inicia el proceso que describimos con anterioridad.



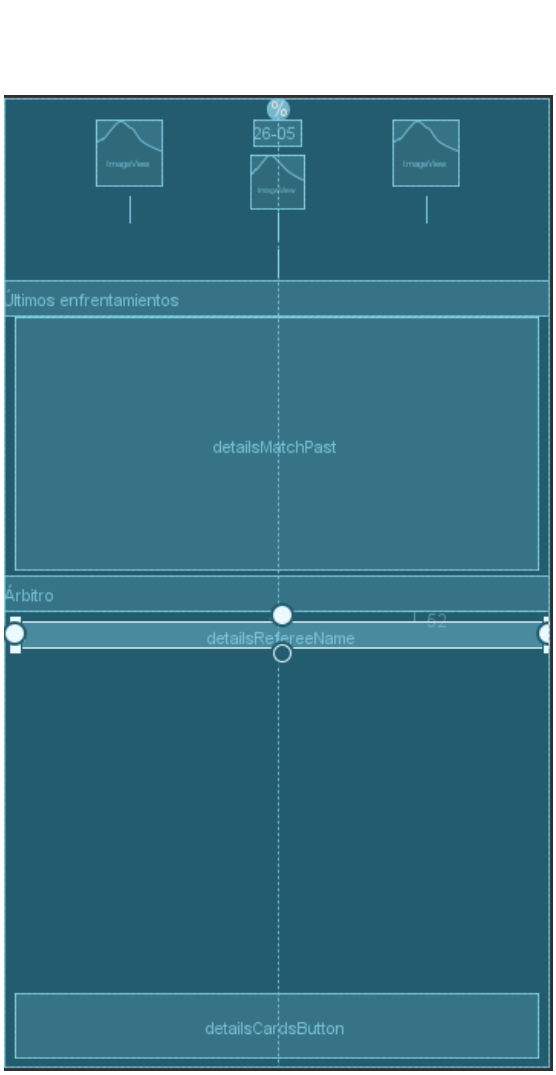
(a) Diseño XML

(b) Diseño gráfico.

Figura A.2: Vista de Partidos.

Vista de Predicción

Una vez el usuario ha elegido un partido de la lista abrimos la ventana con la información relacionada. La información meteorológica se la mostramos de forma gráfica. Puede ver los últimos enfrentamientos entre estos equipos de los que disponemos información. Los objetos seleccionables son los partidos pasados y el botón de estadísticas. Para volver atrás debe usar las teclas navegación de Android.



(a) Diseño XML

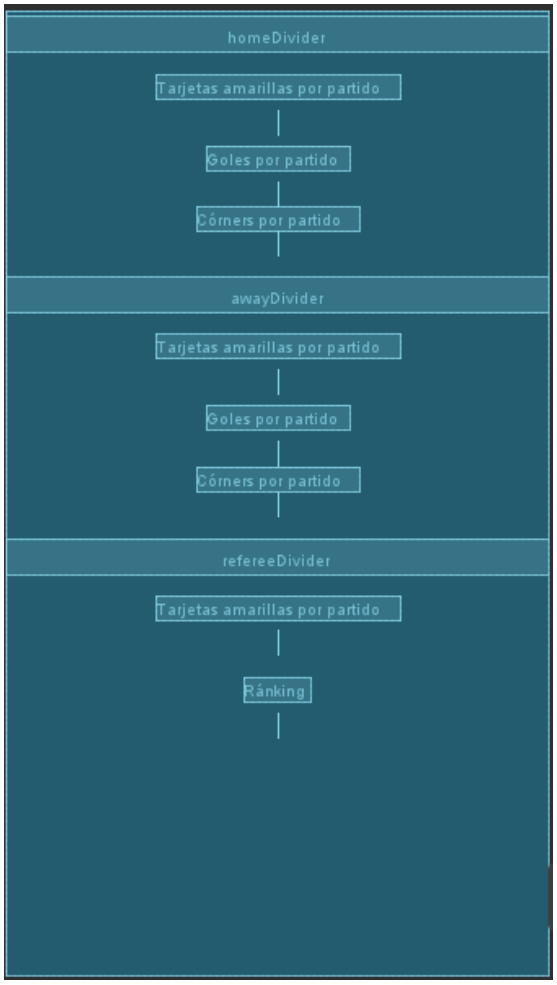
(b) Diseño gráfico.

Figura A.3: Vista de Predicción

Vista de Estadísticas

Si el usuario elige darle al botón de estadísticas es donde mostramos las predicciones para el encuentro.

Comentaremos un caso práctico mas adelante en las pruebas.



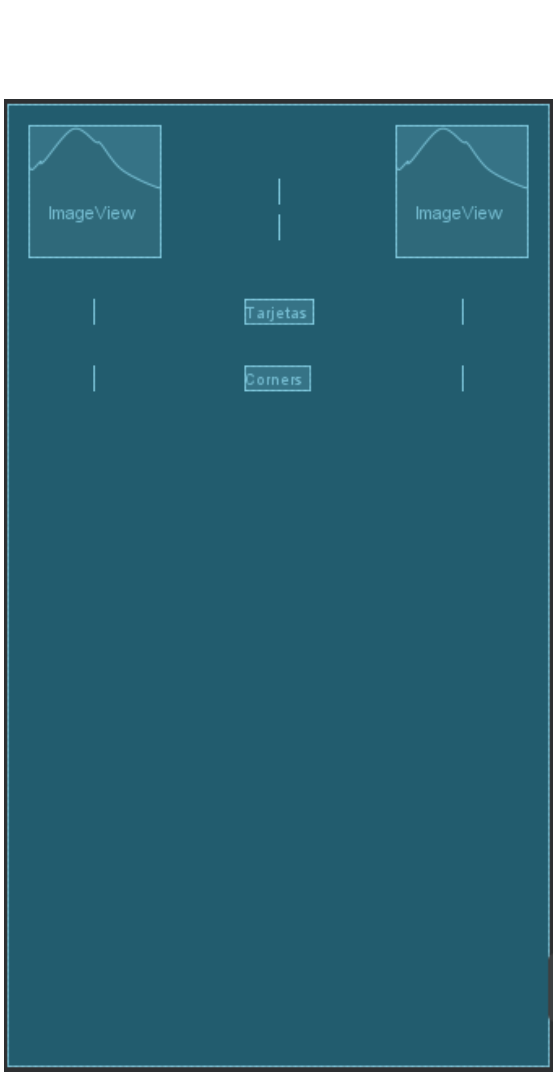
(a) Diseño XML

(b) Diseño gráfico.

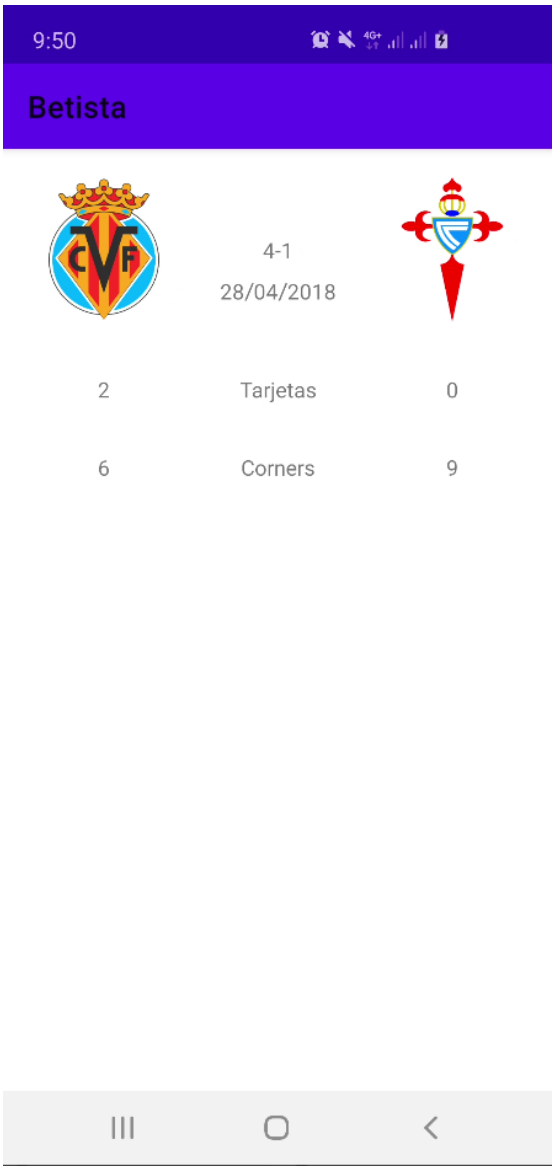
Figura A.4: Vista de Estadísticas.

Vista de Partido Pasado

Finalmente si el usuario eligió algún partido pasado, le mostramos los resultados que se dieron en ese partido.



(a) Diseño XML



(b) Diseño gráfico.

Figura A.5: Vista de Partido Pasado.

Lista de acrónimos

ETL *Extract Transform and Load.*

BD *Base de datos.*

DAO *Data access object.*

DTO *Data transfer object.*

Glosario

Tipster Persona que ofrece de manera gratuita o remunerada pronosticos de apuestas deportivas.

Tip Sugerencia de resultado de apuesta deportiva ofrecida por un Tipster.

RAW Datos en bruto y sin procesar referentes a los partidos, en este tipo de datos la clave primaria no se respeta y existen varías lineas para un solo partido.

BD Base de datos de la APP. Responde en la direccion: tryker.no-ip.org:3306

BROKER Casa de apuestas o persona que acepta tramitar apuestas deportivas en su nombre o en el de otro.

marketplace Término acuñado por John Sviokla que define el sitio en Internet donde se llevan a cabo interacciones comerciales entre diferentes empresas. ... En definitiva, es el lugar en la red donde se llevan a cabo acciones comerciales, igual que en la plaza del mercadillo pero a nivel virtual y online.

FEJAR Federación Española de Jugadores de Azar Rehabilitados. https://fejar.org/

Daemon Servicio o programa residente es un tipo especial de proceso informático no interactivo, es decir, que se ejecuta en segundo plano en vez de ser controlado directamente por el usuario.

Servlet Un servlet es una clase en Java, se utiliza para codificar las respuestas de un servidor ante diferentes peticiones como si de una hoja de ruta se tratase. Son utilizados comúnmente para extender las aplicaciones alojadas por servidores web.

Bibliografía

[1] Mirada21, "España tiene la cifra más alta de europa de jóvenes ludópatas," 2019. [En línea]. Disponible en: http://mirada21.es/secciones/mirada-alternativa/espana-tiene-la-cifra-mas-alta-de-europa-de-jovenes-ludopatas/

[2] U. de Alicante, "Patrones de diseño para aplicaciones locales sin ejbs," 2014. [En línea]. Disponible en: http://www.jtech.ua.es/j2ee/2003-2004/modulos/pd/sesion02-apuntes.htm

[3] MicroStrategy, "Definición ¿qué es el modelado predictivo?" 2020. [En línea]. Disponible en: https://www.microstrategy.com/es/resources/introductory-guides/predictive-modeling-the-only-guide-you-need#:~:text=El%20modelado%20predictivo%20es%20un,avances%20tecnol%C3%B3gicos%20y%20ganancias%20empresariales.

[4] betsson.es, "¿qué son las cuotas en apuestas deportivas y cómo funcionan?" 2018. [En línea]. Disponible en: https://www.betsson.es/blog/tutoriales/apuestas-deportivas/las-cuotas-apuestas-deportivas/

[5] G. de España, "Boletín oficial del estado, nº174," 2019. [En línea]. Disponible en: https://www.boe.es/boe/dias/2019/07/22/pdfs/BOE-A-2019-10740.pdf

[6] R. Italiana, "Artículo 9, gazzett aufficial n. 87," 2019. [En línea]. Disponible en: https://www.gazzettaufficiale.it/atto/serie_generale/caricaArticolo?art.progressivo=0&art.idArticolo=9&art.versione=1&art.codiceRedazionale=18G00112&art.dataPubblicazioneGazzetta=2018-07-13&art.idGruppo=4&art.idSottoArticolo1=10&art.idSottoArticolo=1&art.flagTipoArticolo=0#art

[7] Wikipedia, "Proveedores y patrocionadores real madrid," 2020. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Historia_del_uniforme_del_Real_Madrid_Club_de_F%C3%BAtbol#Proveedores_y_patrocinadores

[8] F. C. Barcelona, "Betfair, nuevo patrocionador fc barcelona," 2016. [En línea]. Disponible en: https://www.fcbarcelona.es/es/noticias/1064520/betfair-nuevo-patrocinador-del-fc-barcelona

[9] M. C. R. Alex Meadows, Adrián Sergio Pulvirenti, *Pentaho Data Integration Cookbook Second Edition*, 2nd ed. Packt Publishing, 2013.

[10] C. Pentaho, "Manual del usuario de spoon," 2010. [En línea]. Disponible en: https://wiki.pentaho.com/display/EAIes/Manual+del+Usuario+de+Spoon

[11] Hitachi, "Pentaho official documentation," 2020. [En línea]. Disponible en: https://help.pentaho.com/Documentation/9.0

[12] MySQL, "Pentaho official documentation," 2020. [En línea]. Disponible en: https://dev.mysql.com/doc/refman/8.0/en/

[13] Kotlin, "Pentaho official documentation," 2020. [En línea]. Disponible en: https://kotlinlang.org/docs/reference/

[14] ORACLE, "Java™ platform, standard edition 8 - api specification," 2020. [En línea]. Disponible en: https://docs.oracle.com/javase/8/docs/api/

[15] U. de Alicante, "Spring," 2020. [En línea]. Disponible en: http://www.jtech.ua.es/j2ee/publico/spring-2012-13/wholesite.pdf

[16] Google, "Pentaho official documentation," 2020. [En línea]. Disponible en: https://developer.android.com/docs

[17] SportMonks, "Sportmonks official documentation," 2020. [En línea]. Disponible en: https://sportmonks.com/docs/

[18] M. M. B. Mónica Martínez Gómez, "La distribución poisson," 2014. [En línea]. Disponible en: https://riunet.upv.es/bitstream/handle/10251/7937/Distribucion%20Poisson.pdf